Bounded Rationality

Bounded Rationality

David C. Parkes
Computer and Information Science Department
University of Pennsylvania
200 South 33rd Street, Philadelphia, PA 19104
dparkes@unagi.cis.upenn.edu

Abstract

Russell and Wefald (Russell & Wefald 1991) propose that the study of resource-bounded intelligent systems should be central to artificial intelligence. A resource-bounded agent acting in a time-critical domain must decide what to reason about, when, and for how long. Too little reasoning can lead to mistakes, while too much can lead to lost opportunities. We use the term bounded rationality to mean reasoning that is optimal in a utility-maximizing sense, given the bounded-resources of the agent, and the dynamic environment within which the agent is situated. Bounded rationality necessarily requires optimal meta-reasoning, that is reasoning about reasoning. This paper surveys recent research into resource-bounded rationality for artificial intelligent agents. The central thesis of this work is that it is not appropriate to design agents using a normative theory of rationality, such as decision-theoretic inference, and then add heuristics to allow the agents to operate in time-dependent environments.

Russell and Wefald present a general framework for decision-theoretic metareasoning, that is reasoning about reasoning. Russell and Wefald use probability and decision theory to develop a general formula for the utility of computation. The utility of a computational action is derived from its effect on the agent's choice of action in the world. Their work is related to some early work in the decision sciences by Good (Good 1971) and Simon (Simon 1976). The problem is cast as search, with the marginal value of computation determining the optimal sequence of computations. Russell and Wefald present an application to competitive game-playing, and are able to demonstrate a substantial improvement in search efficiency when compared to alpha-beta search.

Boddy and Dean (Boddy & Dean 1989), and Horvitz (Horvitz 1987), independently developed an approach to meta-reasoning for a special class of decision procedures, known as flexible computations, or anytime algorithms. Anytime algorithms guarantee that the quality of a solution increases monotonically with time, and can be interrupted at any time. The flexibility simplifies the problem of balancing computation time and quality in an uncertain environment. Boddy and Dean present a class of efficient algorithms for optimal deliberation-scheduling on a set of anytime algorithms that compute the best response of an agent to a series of events in the world. Horvitz presents an application to time-critical medical decision making, and Boddy and Dean present an application to a robot courier domain. The decision-theoretic meta-level of Russell and Wefald can be seen as a means to construct a near-optimal anytime algorithm.

Zilberstein addresses the *composition problem*, that is how to schedule deliberation to a system of

dependent anytime algorithms. The general composition problem is shown to be $\mathcal{NP}$-complete, but solutions are provided for restricted cases. Zilberstein also introduces the idea of *monitors* that allow the resource-allocation strategy to be dynamically adjusted as the computation progresses. An active monitor can be viewed as an extension of the decision-theoretic control proposed by Russell and Wefald, since control decisions are taken on the basis of marginal value.

keywords: bounded rational, decision theory, real-time

1 Introduction

Artificial Intelligence is the discipline that is concerned with programming computers to do clever, humanoid things - but not necessarily to do them in a humanoid way. (Simon 1978)

It seems pertinent to begin our discussion of resource-bounded rationality, and what it means to be intelligent, with the man-machine contest on the 35th floor of the Equitable Center, mid-town Manhattan, that has been dominating the media all week. Gary Kasparov, chess world champion, succumbed to the powers of Deep Blue, the machine designed and trained by researchers and chess experts at I.B.M. Although the timing of this victory was perhaps surprising, its ultimate inevitability was never in doubt.

Many commentators are insisting that Deep Blue has no intelligence whatsoever, because it does not "understand" a chess position, but searches through billions of moves "blindly". But what better indication is there of "understanding" a position than to play the best move? Taking the functionalist position, it seems that the ability of Deep Blue to play chess better than any human is evidence enough of its intelligence. We can view Deep Blue as a black box, able to choose good moves, in just the same way that the brain of a grandmaster is "wired" to produce good moves. The brain works because billions of neurons carry out hundreds of tiny operations each second, none of which in isolation demonstrates any intelligence.

Perfect rationality, that is making *the* best move in response to the move of one's opponent, cannot make a distinction between intelligent behavior that does the right thing *by some means*, and intelligent behavior as the result of intelligent cognition. Perfect rationality is judged solely on the external actions of the agent in the world. But when we have bounded-resources we cannot really judge an agent that fails to make the best move as irrational. We need a weaker version of rationality that judges an agent not only

only the actions that it takes, but also by the reasoning it uses and the resources that are available. We use the term "bounded rationality" to mean reasoning that is optimal in a utility-maximizing sense, given the bounded-resources of the agent, and the dynamic environment within which the agent is situated. Bounded rationality necessarily requires optimal meta-reasoning, that is reasoning about reasoning. The reasoning process becomes important precisely when we have bounded-resources. The game of chess has been estimated to have at least 10^{120} possible paths: a search space which will always make artificial agents resource-bounded.

There is good evidence that grandmasters compare up to 50,000 board positions that they have studied to the current situation, and that they are able to choose the right move "intuitively", because they are *unaware of the reasoning process that is occurring in the brain* (Chase & Simon 1973). If the computer did the same thing the trick would be revealed - but would it not still be "intelligent"? The metareasoning that happens unconsciously in the brain must be explicitly considered in an artificial agent.

We also need the concept of resource-bounded rationality for the *design* of intelligent artificial agents. The work on the decision-theoretic control of computation that we survey here seeks to design systems that account for their own bounded computational resources during problem solving. Deep Blue can analyze 300 million chess positions in a second, and is also endowed with expert knowledge to guide its search. The importance of good search control is highlighted by what it still cannot do. During the third game in the series Kasparov played a bold pawn sacrifice that I.B.M. researchers said Deep Blue would never have made because "it gave too much away, too quickly". Kasparov was able to use pattern matching and intuition to recognize the move, while Deep Blue would have needed to do a very deep search to recognize its value. The combinatorial explosion of the search domain for chess makes the control of search critical. Russell and Wefald (Russell & Wefald 1991) apply decision-theoretic principles to the control of search, and are able to show significant gains over a heuristic control technique such as alpha-beta pruning for competitive-game playing.

A resource-bounded agent acting in a time-critical domain must decide what to reason about, when, and for how long. Too little reasoning can lead to mistakes, while too much can lead to lost opportunities. This paper surveys recent research into resource-bounded

rationality for artificial intelligent agents. The central thesis is that it is not appropriate to design agents using a normative theory of rationality, such as decision-theoretic inference, and then add heuristics to allow the agents to operate in time-dependent environments. Russell and Wefald (Russell & Wefald 1991) propose that the study of resource-bounded intelligent systems should be central to artificial intelligence.

1.1 A brief history of rationality within AI

In artificial intelligence the logical approach provided the first formal definition of rationality (McCarthy 1958). Logical rationality assumes that an agent can be described in terms of its beliefs and goals, and that an agent is rational if it satisfies one of the goals entailed by its beliefs. The emphasis within AI in the early days was on normative reasoning, that is on giving a consistent set of rules under which the agent could reason. Formal intractability results in computation were unknown, and the aim was to reason using a consistent set of axioms, and generate provably correct plans. The emphasis was on finding a *satisficing* solution, that is any solution to a problem that solved the goal, and there was no notion of quality. The approach is summarized by Newell:

> *If one of the available actions leads to one of the goals, take it!* (Newell 1981).

Another view of rationality is termed *economic rationality* (Doyle 1989). Economic rationality provides formal tools for understanding heuristics and reasoning, and a rigorous normative framework for analyzing utility and probability. We use utility theory to represent degrees of goodness: an agent is rational if it chooses the action with the maximum expected utility. Economic rationality can be viewed as generalizing the logical approach to allow for goal conflicts and uncertainty. Instead of competing as normative theories, logical and economic notions of rationality fill complementary needs. Logic describes the possibilities for reasoning and action, and economics allows an agent to make choices among these. However, decision-theoretic reasoning is in general $\mathcal{NP}$-hard (Cooper 1990). The normative use of decision theory provides a standard for rationality, but one which is often unattainable due to limitations on the available information or resources.

There have been two main approaches to solving the problem of what a bounded-rational agent should do do in real-time domains, when the ability to trade-off quality and speed of response is critical. The first approach is known as *reactive* reasoning (Agre

& Chapman 1991; Brooks 1986). Explicit reasoning, problem solving and maintaining a world model are abandoned. Instead agents have reactive mechanisms that generate actions in response to interactions with the physical world. Brooks argues that we can achieve intelligence in this way *without reason*, that agents can be right by design. A similar approach is that of *Universal Planning* (Schoppers 1987). Agents determine what to do next by finding a result in a large look-up table. This solution to the problem of bounded-resource agents in time-critical environments seems unsatisfying, and is also infeasible in complex domains where the look-up table quickly becomes too large. Reactive systems provide responsiveness at the cost of inflexible and shallow decision making. It seems inconceivable that such an approach could ever be used for complex medical domains, path-planning, or chess. The approach is not very useful either, as it transfers the complexity to the design phase, and ignores the complexity of the designer (Zilberstein 1993).

The second approach is to add heuristics to normative reasoning methods in an attempt to reduce the complexity of the problem. This approach can be criticized as inadequate because it makes little sense to design a normative decision system only to arbitrarily add heuristics to make it useful to a resource-bounded agent. Most heuristic methods are thought to increase utility, but are used without any real information about their probability of usefulness (e.g., A* search, constraint propagation, alpha-beta pruning, etc.).

With resource-bounded agents we need a new model of rationality, that judges agents not only by their actions in the world, but also by the reasoning process by which they choose those actions. We must consider the reasoning process in our concept of rationality, without that we are left with an inadequate *theory* for designing intelligent systems. Consider the haphazard logical approach of deducing new conclusions until termination (success), or a deadline is reached(failure), or the decision-theoretic approach that assumes that we are able to compute NP-hard probabilistic reasoning results, and considers an agent as irrational if it does not take the best action.

The new approach presented here uses a normative analysis at the *meta-level* to control reasoning in real-time environments for resource-bounded agents. The goal of this work is to provide a rational approach to rational reasoning under scare resources. Rationality for a resource-bounded agent differs from the pure decision-theoretic notion of rationality, in that it explicitly allows for finite computation resources. The value of a decision is judged in

terms of the effect that it has on actions performed by the agent, noting that both actions and computation have time value.

1.2 Views from the Decision Sciences

Early discussion of bounded rationality within the decision sciences presumed that a normative approach to metareasoning would be too costly. Good (Good 1971) provided the earliest discussion of the explicit integration of the costs of inference within the framework of normative rationality. Good made a distinction between classical, or "Type I" rationality, and what he called "Type II" rationality, or the maximization of expected utility taking into account deliberation costs. "Type I" inference is consistent with the axioms of decision theory without regard to the cost of inference, and "Type II" is behavior that takes into consideration the costs of reasoning. "Type I" agents are judged as rational if the results satisfy their preferences, "Type II" agents are judged as rational if they use their limited resources optimally with regard to their preferences.

Simon (Simon 1976) also considers two types of rationality: *procedural rationality* where the agent must *compute* the rational thing to do, and *substantive rationality* where the agent must simply *do the right thing*, and rationality is judged by actions in the world. Substantive rationality depends only on the preferences of the agent, while procedural rationality depends both on the preferences of the agent, and the process by which it reasons. Simon notes that the shift of emphasis from substantive to procedural rationality, that he advocates, also requires a shift from concern for optimal solutions to a concern for good solutions. Simon uses chess as an example that human behavior is not substantively rational, but procedurally rational. Procedural rationality of a grandmaster in chess comes from heuristics for selective search, and knowledge of significant patterns. Search is terminated when a satisfactory solution is found, not when the optimal solution is found (Newell & Simon 1972).

The theory of information value (Howard 1966) has many parallels with the work on metareasoning that we survey here. Howard proposes the idea of computing the decision-theoretic value of an additional piece of information by simulating the decision procedure that will follow, given each possible outcome of the information request, and thereby estimating the expected value of the information.

1.3 Rationality and Intelligence

Russell (Russell, Subramanian, & Parr 1993; Russell 1995) outlines four candidates for the formal definition of intelligence. The first is *perfect rationality*, which defines rationality as the ability to perform the action with the best expected utility given available information about the environment. This is Good's "Type I" rationality, and Simon's "substantive rationality". The second is *calculative rationality*, which is the ability to achieve perfect rationality if the decision process is executed infinitely fast. This has been the main focus of much research in AI: it concentrates on "epistemological adequacy" before "heuristic adequacy". Logical planning systems using situation calculus, and systems based on influence diagrams (belief nets with embedded actions), are calculatively rational. The third definition is *meta-level rationality*, which views rationality as the ability to take rational decisions at the meta-level, this is what Good meant by "Type II" rationality. A meta-level rational agent selects computations according to the expected utility. The fourth is *bounded optimality*, which is a term originated by Horvitz (Horvitz 1987). Bounded optimality is really the ultimate goal for a real artificial intelligent agent. A decision process is bounded optimal if the expected utility of the actions it recommends is at least as high as any other decision process with the same resource-bounds, and in the same environment.

2 Russell and Wefald: Decision-theoretic Control of Inference

Russell and Wefald present a general framework for the control of reasoning for a resource-bounded rational agent. A resource-bounded agent in a time-critical environment must necessarily perform metareasoning, with a view to using its computational resources on the base-level reasoning that has the greatest expected utility. In a uniform approach, Russell and Wefald propose that the meta-level problem, of reasoning about reasoning, should be cast in the same language as the base-level problem, or reasoning about actions. A bounded-rational agent should take the computational actions that have the highest expected utility. Russell and Wefald view the value of a computation as deriving from the effect that the computation has on the actions that an agent chooses to take in the world. Computation refines the beliefs that an agent has about the value of an action. Russell and Wefald correctly note that reasoning about a computation without doing it necessarily involves uncertainty as to its outcome, so probability and decision theory are the

main tools of normative metareasoning. Probability and utility theory provide a normative basis (accepted in several disciplines as the right thing to do, a consistent axiomatic basis that is optimal) for a *rational* theory of belief and action. Russell and Wefald present normative methods to control metareasoning. The base-level problem is cast as search, and the principles of metareasoning propose that the marginal value of expanding a node should be the basis for search control. Expanding a node in a decision tree corresponds to refining the assessment of utility that an agent assigns to the base-level actions.

In general, meta-reasoning itself is costly, and a normative approach requires that we also choose the amount of deliberation to spend on meta-reasoning, with an explicit assessment of its expected value to the agent. There is a clear problem of infinite regress when suggesting normative control at all levels of the agent. In practice, the deliberative information available at lower levels decreases rapidly, and most systems rely on well-chosen default actions at the first or second levels. Russell and Wefald make a number of simplifying assumptions at the meta-level, and claim that the costs of metareasoning are small enough to ignore.

2.1 A general framework of metareasoning

Let Ω denote the set of possible states of the world. A state of the world includes the environment (possibly dynamic) and the agent's internal state (beliefs). The agent has a utility function that expresses its preferences over states in the world.

$$U : \Omega \rightarrow [0, 1] \tag{1}$$

An agent prefers state ω_2 to ω_1, written $\omega_1 \prec \omega_2$, precisely when $U(\omega_1) < U(\omega_2)$. The agent has a set of possible base-level actions, $\mathcal{A}$, that affect its environment, and at any time an action, $\alpha \in \mathcal{A}$, that is estimated to transform the current state to a new state with the highest expected utility. The other actions that the agent might choose are denoted $\{\beta_1, \ldots, \beta_n\}$, where β_1 is the current second-best action for the agent.

The agent also has a set $\{S_i\}$ of computational actions which refine the estimate of value that an agent assigns to each base-level action. We denote the outcome of an action (computational or base-level) X, performed in state ω, by $[X, \omega]$ or just $[X]$ if the action is performed in the current state. Russell and Wefald consider the case where each action has a deterministic outcome.

Let $\mathbf{S}$ denote a sequence of computational actions, and $\mathbf{S}.S_j$ a sequence of actions $\mathbf{S}$ followed by action S_j. The meta-level decision facing the agent is whether to perform a computational action S_j, or take the current best base-level action α. The value of a computation derives from any improvement in base-level action due to the refinement of the values assigned to individual actions.

We first assume that the computations $\{S_i\}$ are *complete*, that is no further computation will take place following any of the computations. The utility associated with a complete computation S_j in the current state is defined as the utility of the state that results from taking the new assessment of best action, α_{S_j}, from the state resulting from the computation, $[S_j]$.

$$U([S_j]) = U([\alpha_{S_j}, [S_j]]) \tag{2}$$

In general, the computation may be part of a sequence of computations, and the utility of the action is defined as the expected utility of the base-level action that the agent will *finally take*, given the computation sequences $\mathbf{T}$ that the agent might take as a result of computation S_j:

$$U([S_j]) = \sum_{\mathbf{T} \in \mathcal{T}} \Pr(\mathbf{T}) U([\alpha_{\mathbf{T}}, [S_j.\mathbf{T}]]) \tag{3}$$

where $\Pr(\mathbf{T})$ is the probability that the agent will perform the computation sequence $\mathbf{T}$ subsequent to S_j. The marginal value of a computation is given by the difference in utility between the state that results from the computation, and the current state. The utility associated with the current state is just the utility of taking the current best action, α, from that state.

$$V(S_j) = U([S_j]) - U([\alpha]) \tag{4}$$

The marginal value of a computational action computed here is very general, and allows for the cost of deliberation in the real world (since the utility of the action after computation S_j is explicitly conditioned on the state after the computation, while the utility of action, α, is conditioned on the state before the computation). We can reason *qualitatively* about when further reasoning is valuable. Consider an agent that is faced with a choice between two actions in the world. The agent currently has an estimate of the utility of each action, but the estimates are subject to some uncertainty. Figure 1 illustrates three possible probability

distributions for utilities assigned to the actions. In (a) the agent should stop reasoning, because the possibility of a change in action preference with any reasonable amount of computation is negligible. In (b) the agent should also stop reasoning, because although the agent cannot be sure which is best, the current utility estimates are close and the uncertainty small enough that further computation can be expected to reveal no significant difference between the two actions. In (c) there is considerable uncertainty, and considerable overlap in the utility estimates for each action, and further deliberation is recommended.

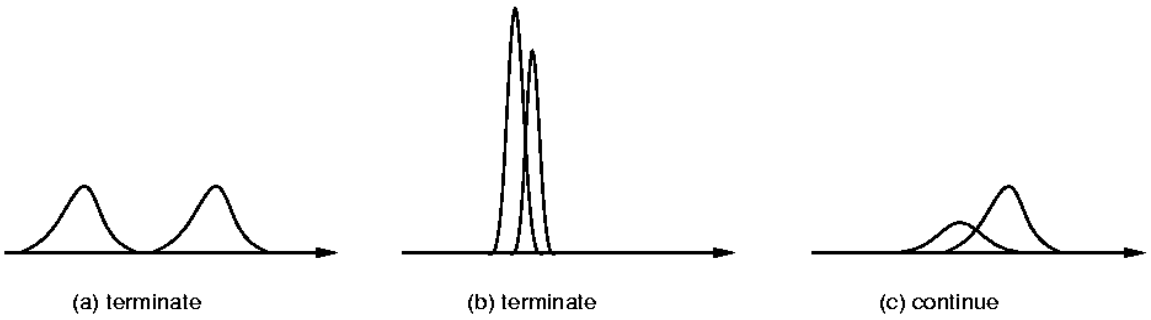

Figure 1: Termination condition using rational metareasoning

2.2 Normative metareasoning for bounded-rational agents

In general the agent does not know the exact utilities or probabilities at any stage in its computations, it can only compute estimates. Let $\hat{Q}^{\mathbf{S}}$ denote the estimate of quantity Q following computation $\mathbf{S}$. At each point in time the agent decides how to act based on the estimates that it currently has, using a meta-reasoning decision procedure whose time cost is considered negligible. The meta-reasoning decides on further deliberation on the basis of the estimated value of computation. The value of computation (equation 4) becomes:

$$\hat{V}^{\mathbf{S},S_j}(S_j) = \hat{U}^{\mathbf{S},S_j}([S_j]) - \hat{U}^{\mathbf{S},S_j}([\alpha]) \tag{5}$$

There is a subtle problem with introducing estimates of the utility of states into the analysis. The net value of a computation must be judged relative to the (possibly revised) estimate of the utility of taking action α in the current state. We need $U([\alpha]) = \hat{U}^{\mathbf{S},S_j}([\alpha])$, and not $U([\alpha]) = \hat{U}^{\mathbf{S}}([\alpha])$. Consider what would happen if the computational action S_j revises only the estimate of value that we assign to our current best action, α. Clearly, when

$\alpha_{S_j} = \alpha$, we require that $V(S_j) \leq 0$, since we will take the same action in the world (with the same value), and the time of deliberation might be costly. Also consider the case where $\hat{U}^{\mathbf{S}.S_j}([\alpha]) > \hat{U}^{\mathbf{S}}([\alpha])$. We do not want to view this as a beneficial computation because we will still choose to take action α in the world, and the true utility of action α in the world is unchanged. This is why we calculate the marginal value of computation with respect to the *revised* value for α. Similarly, we could have the case of a revision of value for α to below that of the current second-best action β_1. This should have a positive value, and never a negative value.

To simplify reasoning about the utility of combined computational and base-level actions, Russell and Wefald separate the *intrinsic utility*, the utility of an action independent of time, from the *time cost* of computational actions. The utility of taking an action A_i in a state resulting from computation S_j is written:

$$U([A_i, [S_j]]) = U_I([A_i]) - C(A_i, S_j) \tag{6}$$

where $U_I([A_i])$ represents the intrinsic utility of the action A_i, relative to the current state, and $C(A_i, S_j)$ represents the cost associated with delaying action A_i by computation S_j. Russell and Wefald assume that $C(A_i, S_j) = C(S_j)$, so that computations have the same effect on the utility of all actions. Thus is approximately true in many AI domains, such as path-planning in a slowly changing environment, but not in domains where the utility of different actions decreases at different rates, such as deciding between catching the bus, cycling, or walking. Russell and Wefald also assume that the cost of a computation depends only on the elapsed time, and write $C(A_i, S_j) = \mathrm{TC}(|S_j|)$. With this in hand, we can separate out the cost and benefit of a computation, and equation 5 becomes

$$\begin{aligned} \hat{V}([S_j]) &= \hat{U}^{\mathbf{S}.S_j}([\alpha_{S_j}, [S_j]]) - \hat{U}^{\mathbf{S}.S_j}([\alpha]) \\ &= \hat{U}_I^{\mathbf{S}.S_j}([\alpha_{S_j}]) - \hat{U}_I^{\mathbf{S}.S_j}([\alpha]) - \mathrm{TC}(|S_j|) \end{aligned} \tag{7}$$

The agent cannot know the values associated with $\hat{U}_I^{\mathbf{S}.S_j}([\alpha_{S_j}])$ or $\hat{U}_I^{\mathbf{S}.S_j}([\alpha])$ without performing the actual computation, and metareasoning must use a statistical analysis to control reasoning. The agent also needs a model of the environment to be able to determine how the would will change during computation S_j. The value of the computation

S_j is a random variable before computation S_j is performed, and the agent computes an expectation

$$E[\hat{V}^{\mathbf{S},S_j}(S_j)] = E[\hat{U}^{\mathbf{S},S_j}([S_j])] - E[\hat{U}^{\mathbf{S},S_j}([\alpha])] \tag{8}$$

2.3 An application to competitive game-playing

Russell and Wefald make a number of simplifying assumptions at the meta-meta level to enable an application of the principles of metareasoning outlined above to a real problem. The assumptions are necessary so that the cost of meta-level reasoning level are small. The myopic assumptions have already been touched upon:

- **Single-step assumption**. Russell and Wefald view each computational action as a *complete* computation: the value of a computation step is estimated as though it is the last computation that will ever be executed before a real-world action is chosen. This gives an inaccurate estimate of the value of a computation. Consider the case of a computation step that has no immediate value, but is the preliminary step to a computation that will significantly improve the values that the agent assigns to base-level actions. The single-step assumption is *myopic*, and would recommend against the computation. The policy is termed single-step because it assumes that the agent will take only one more computation.

- **Meta-greedy assumption**. Russell and Wefald only consider a choice between single computational actions, since it is intractable to consider all possible computational *sequences* that the agent could follow. The policy is termed meta-greedy, because it amounts to a greedy choice of the next computation step at the meta-level.

We need *both* assumptions. If we relax the meta-greedy assumption and consider *sequences* of computations then it is no constraint to view the computations as complete. Similarly, if we accurately compute the full expected value of a single computation step (including the effect on future computations), then considering only the next computation step is not a constraint.

Russell and Wefald apply decision-theoretic search control to competitive game-playing and single-agent problem solving. The computation is readily divided into units by considering each computation step as expanding a node in a search tree, applying an evaluation

function at that node, and then back-propagating the value up the tree to base-level actions at the root. Statistics are used to estimate the net value of an expansion in improving the utility estimate assigned to a base-level action, and search control proceeds using marginal value as the decision criteria. The agent refines the estimates that it has for the utility of its available actions, and deliberation proceeds until no computation has expected positive marginal value, at which point the best base-level action is chosen. An efficient search algorithm, MGSS*, is developed for Othello that soundly beats heuristic alpha-beta search in a tournament of games, while expanding significantly fewer nodes and taking less time (Russell & Wefald 1991).

A search program proceeds by expanding leaf nodes and propagating the value information back to the parents. The value that will be assigned to a new leaf node depends on the type of leaf node and the nature of the computation. Russell and Wefald statistically estimate the *local effect* of a computation since it depends on the nature of the node being expanded, and the nature of the expansion computation. The effect of the expansion on the other nodes in the tree, and in particular on the one relevant base-level action, is then an analytic function of the current state of the tree (the current value estimates for the nodes on the path to the root, and their children).

Russell and Wefald assume *subtree independence*: a computational action can affect the estimated utility for exactly one base-level action. A computation is beneficial if (1) we come to prefer a new external action β_j over the current preferred action α, or (2) computation on α causes its utility estimate to be revised below that of the current second-best move. Suppose that we are considering the computation S_j, which affects only the estimated utility of action β_j. The computation will only change our choice of base-level action if $\hat{U}_I^{\mathbf{S}.S_j}([\beta_j]) > \hat{U}_I^{\mathbf{S}}([\alpha])$. The intrinsic utility assigned to action α is assumed to be unaffected by computation S_j (subtree-independence). The expected value of the computation step S_j is therefore:

$$E[\hat{V}(S_j)] = \int_{\hat{U}_I^{\mathbf{S}}([\alpha])}^{1} p_{\beta_j,j}(x)(x - \hat{U}_I^{\mathbf{S}}([\alpha]))dx - \mathrm{TC}(|S_j|) \tag{9}$$

where $p_{\beta_j,j}(x)$ is the probability density function for $\hat{U}_I^{\mathbf{S}.S_j}([\beta_j])$. The intuition behind this expression for the expected value of computation S_j is that we will take the action β_j only when $\hat{U}_I^{\mathbf{S}.S_j}([\beta_j]) > \hat{U}_I^{\mathbf{S}}([\alpha])$, and otherwise (for $x \leq \hat{U}_I^{\mathbf{S}}([\alpha])$ we take the existing action, α,

and the computation has no value.

Similarly, if we perform a computation S_k, that affects only the utility estimate of the current best action α, then we will choose the current second-best action β_1 if $\hat{U}_I^{\mathbf{S}}([\beta_1]) > \hat{U}_I^{\mathbf{S}.S_k}([\alpha])$, and the expected value of computation step S_k is:

$$E[\hat{V}(S_k)] = \int_0^{\hat{U}_I^{\mathbf{S}}([\beta_1])} p_{\alpha,k}(x)(\hat{U}_I^{\mathbf{S}}([\beta_1]) - x)dx - \mathrm{TC}(|S_k|) \tag{10}$$

where $p_{\alpha,k}(x)$ is the probability density function for $\hat{U}_I^{\mathbf{S}.S_k}([\alpha])$. The intuition is that we only take the old second-best action if $\hat{U}_I^{\mathbf{S}.S_k}([\alpha]) < \hat{U}_I^{\mathbf{S}}([\beta_1])$, and otherwise for $x \geq \hat{U}_I^{\mathbf{S}}([\beta_1])$ we take the existing action, α, and the computation has no value.

With the relevant probability distributions, evaluation functions, and time cost function, $\mathrm{TC}(|S_j|)$, we can use these expression to efficiently choose the best computational action to take next. If there are n computational actions and m base-level actions, then each meta-level reasoning step requires mn evaluations each of the two above equations. If the distributions for utility estimates are in simple form (parameterized distributions) then we can calculate the results quite efficiently.

2.4 A Othello tournament: MGSS* vs. alpha-beta

Russell and Wefald derive a formula for the value of expanding a leaf node that can be computed with very little overhead given the simplifying assumptions outlined above (Russell & Wefald 1989). The search algorithm was implemented for *Othello*, and Russell and Wefald played five 32-game tournaments against an alpha-beta algorithm with depth limits from two to six; both algorithms used the same evaluation function. The statistical data used to estimated the probability distributions was collected on roughly 35,000 data points. The distribution of the expected value of node expansion was found to be approximately normal, which allowed efficient evaluation of equations 9 and 10. The results, in terms of games won, nodes searched, and CPU time used are given in table 1 below.

Decision-theoretic search control is shown to be up to thirteen times more effective than alpha-beta pruning (it plays slightly better than an alpha-beta search of depth 6 but expands thirteen times fewer nodes). The MGSS* search-algorithm is able to hold its own against an alpha-beta search of depth 6, despite using a myopic metareasoning strategy. Computing expected values for search steps is faster than computing the static evaluation function, enabling the costs of metareasoning to be negligible in comparison

algorithm	*wins*	*nodes*	*time(sec.)*
MGSS*	24.5	3.6K	40
$\alpha - \beta[2]$	7.5	2.4K	23
MGSS*	22	6.0K	68
$\alpha - \beta[3]$	10	8.9K	82
MGSS*	20	12.0K	170
$\alpha - \beta[4]$	12	42.0K	403
MGSS*	16.5	20.7K	435
$\alpha - \beta[5]$	15.5	130.0K	1356
MGSS*	17	44.1K	1590
$\alpha - \beta[6]$	15	567.8K	6863

Table 1: Summary of results for Othello

to the efficiency-gains from expanding significantly fewer nodes. The main problem with MGSS* is the single-step assumption, and this will prevent all node expansions beyond a certain depth. As the MGSS* search tree grows larger it is highly likely that it will reach a situation where no *single* node expansion can alter the choice of best move choice, and the algorithm must conclude that no further search is worthwhile. Russell and Wefald call this the "meta-greedy barrier". Russell and Wefald consider extending the algorithm to allow sequences of steps to be evaluated, and also suggest using MGSS* to selectively extend a depth-8, say, alpha-beta search.

2.5 Russel and Wefald: Summary

Russell and Wefald present a general framework for the control of reasoning for a resource-bounded rational agent. When the model of an agent's decision process that they propose is valid they show that the principles of normative metareasoning can solve the basic problems of real-time AI. Russell and Wefald demonstrate the efficiency gains that can be achieved with meta-level rationality.

The applicability of the decision-theoretic control of search to general reasoning requires that decision-processes can be represented as a discrete sequence of uninterruptible chunks of computation. This is clearly valid for domains like game-playing, where the reasoning process is the expansion of nodes in the game tree, but less valid to continuous domains such as medical decision making. Russell and Wefald also suppose that the time-cost of reasoning can be represented as a simple function of the length of each computational step - this requires fairly strong assumptions about the problem domain, such as that the value

of all actions are subject to the same cost discounting. The time cost may also be hard to determine in more complex domains, and the framework makes it hard to reason about deadlines.

Russell and Wefald introduce heuristics at the metalevel to avoid the need for an infinite regress of metareasoning. The meta-greedy and single-step assumptions that they make seem necessary to make the problem tractable, but also introduce myopia at the meta-level. Russell and Wefald talk about a meta-greedy barrier, and propose that the theory should be extended to allow limited sequences of computational actions to be evaluated. With the metalevel assumptions in hand, Russell and Wefald assume that the cost of metareasoning can be neglected, again this is not reasonable in complex domains.

3 Boddy and Dean's Deliberation Scheduling

The work of Boddy and Dean (Boddy & Dean 1989; 1994) can be viewed as a refinement of the general framework for metareasoning presented by Russell and Wefald. Boddy and Dean study the problem of metareasoning for a specific class of algorithms that have useful properties for real-time planning with resource-bounded agents. The class of algorithms are called *anytime algorithms*. Anytime algorithms are defined as algorithms that return some answer for any allocation of computation time, and are expected to return better answers when given more time. The metareasoning problem is then cast as one of *deliberation scheduling*: a resource-bounded rational agent should allocate computational resources to the anytime algorithm that has the highest utility.

Boddy and Dean (Boddy & Dean 1994) study deliberation scheduling for time-dependent planning problems, when there are events in the real world that the agent must react to, and these events are known ahead of time. As time passes the agent must make a sequence of time-critical decisions. The solution to the deliberation scheduling problem that Boddy and Dean propose is optimal when the events in the world are deterministically known, there is a single anytime algorithm to reason about a response to each event, each anytime algorithm exhibits *decreasing-returns*, and the anytime algorithms are independent. The robot-courier problem that Boddy and Dean consider (Boddy & Dean 1989; Boddy 1991) is representative of a more difficult class of deliberation scheduling problems. In this class of problems the agent is not responding to events in the world, but has a set of tasks to complete as quickly as possible. There are no deadlines, and the agent may reason

about the best way to do a future task while performing its current task. Boddy and Dean are not able to present a general solution to this problem. An efficient algorithm is presented for the robot-courier problem, which is a special case where the anytime algorithms for each task are simple and have the same performance profiles.

Anytime algorithms expand on the traditional view of a computational procedure, where the goal has been on generating an optimal solution as quickly as possible. Anytime algorithms are inherently more flexible: they have the ability to make use of any amount of time, and they are robust to unexpected interruption. Some familiar anytime algorithms include: search techniques such as iterative deepening; various greedy algorithms; iterative methods such as Newton's method; and randomized algorithms such as Monte Carlo algorithms.

Deliberation scheduling, as proposed by Boddy and Dean, is really off-line metareasoning, that requires a deterministic model of the world, and uses the expected performance of anytime algorithms to choose a reasoning strategy that maximizes the expected utility of the agent in the world. Boddy and Dean introduce the idea of a *performance profile* (PP) for an anytime algorithm. This profiles the variation of intrinsic utility of the result of a decision process against run-time. Some typical performance profiles are illustrated in Figure 2.

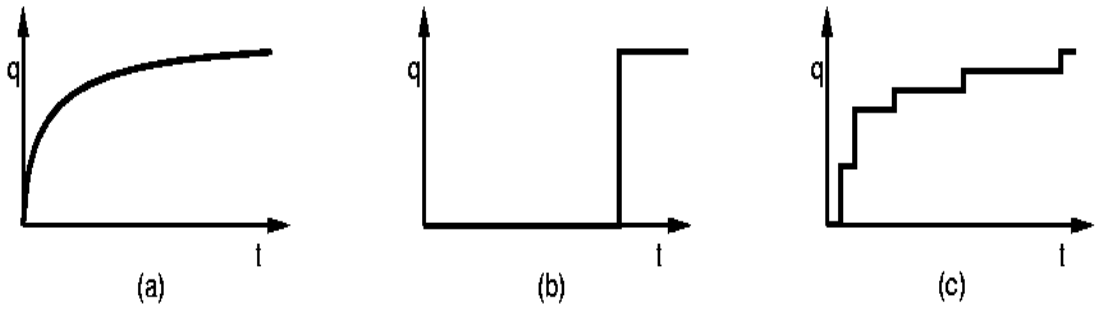

Figure 2: Typical Performance Profiles

Boddy and Dean assume that the goal is to maximize the sum of values of responses to a series of events, the timing of which is known in advance. The problem of optimal scheduling to a sequence of algorithms is $\mathcal{NP}$-hard in the general case, since it subsumes general scheduling problems. However, the deliberation scheduling problem for anytime algorithms is not $\mathcal{NP}$-hard because the anytime algorithms can be suspended and restarted.

This simplifies the deliberation scheduling problem considerably.

3.1 Application: The Robot-Courier Problem

Boddy and Dean (Boddy & Dean 1989; Boddy 1991) use anytime algorithms to solve the problem of a robot courier, that must respond to delivery requests in real-time. Boddy and Dean assume that the requests are deterministically known beforehand, and the problem is to schedule deliberation so as to minimize the *total* time taken to complete all of the tasks. The robot is able to deliberate about a later task as it performs its current task, but Boddy and Dean assume that once the robot starts executing a task it can make no further refinements. The robot has a set of locations that it must deliver packages to within a known world. The world is a grid, and each point on the grid may be occupied by either the robot or an obstacle.

The robot has two anytime algorithms to reason with. The first is a *tour-improvement* algorithm for computing an order in which to visit the locations, known as a *tour*. The tour improvement algorithm generates a tour $\mathcal{S} = \langle l_1, \ldots, l_m \rangle$: an ordered sequence of locations in $\mathcal{L}$, the set of locations that the robot must visit. The algorithm for tour improvement is based on an iterative-improvement algorithm for the Traveling Salesperson Problem (Lin & Kernighan 1973). The algorithm guesses an initial tour, and then produces tours that are progressively closer to an optimal tour by exchanging small sets of edges such that the tour is still complete, but the length of the overall tour decreases (Figure 3). The tour-improvement algorithm has no knowledge of the length of the actual path that the robot will take between consecutive locations, and uses the Manhattan distance between successive locations as a lower-bound on the distance (ignoring possible obstacles).

The second algorithm that the robot may use is a *path-planning* algorithm. The path-planning algorithm plans the exact path that the robot will take between consecutive locations on its tour. The algorithm is a heuristic A* search (Korf 1988), that does not guarantee to generate an optimal path between any two locations, but does better than the dead-reckoning, bouncing off obstacles and finding its way around them, that the robot must otherwise use.

Boddy and Dean assume that there is no advantage in starting until the first location is known. An optimal deliberation schedule for a given situation will be a sequence of tour-improvement and path-planning computations, along with movements in the world, that

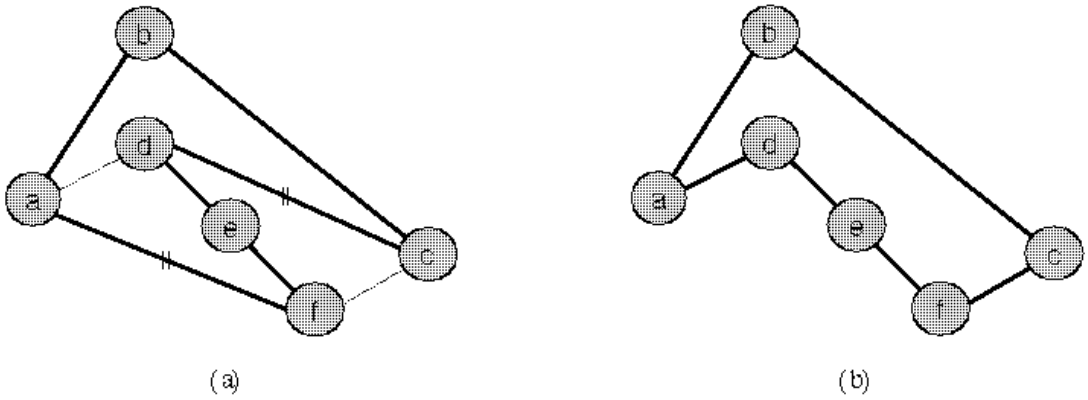

Figure 3: Tour Improvement algorithm

minimizes the total time spent in deliberation and action.

3.2 Generating Performance Profiles

The performance profile of an anytime algorithm plots the expected output quality of the algorithm against run-time. Boddy and Dean (Boddy & Dean 1989) collect statistics on the performance of the algorithms on grid-worlds of a fixed size, with a probability of (.2) that any given location is occupied. A simple grid-world is illustrated in Figure 4.

The performance profile of the tour-improvement algorithm is approximated by a function $f(k) = 1 - e^{\lambda k}$, where λ depends on the size of the tour, and k is the number of edge exchanges made. The value of λ is statistically estimated for the grid-world. A performance profile, showing the expected improvement in tour length as a fraction of the maximum possible, given an allocation of deliberation time δ is shown in Figure 5 (ii).

The path-planning algorithm is an A* search, that uses the distance from the destination to guide the search. The nodes in the search tree correspond to locations in the grid-world, and paths from the root to leaves of the tree correspond to paths through the world. The decision on which node to expand next is taken on the basis of the distance between the node and the destination. The planning procedure terminates when it obtains a complete path. No attempt is made to find an optimal path. A partial path can be returned at anytime by returning the path to the node that is closest to the destination, and letting the robot find its way from there by dead-reckoning. Analysis shows that the algorithm finds a complete path from l_i to l_j in time $\delta_j^* = 1.33\tau_{pp}d_{i,j}$, where τ_{pp} is the time for a single

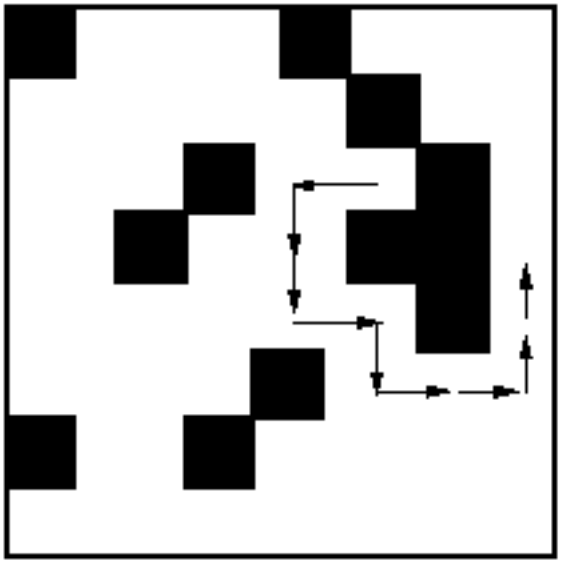

Figure 4: A Simple Grid World

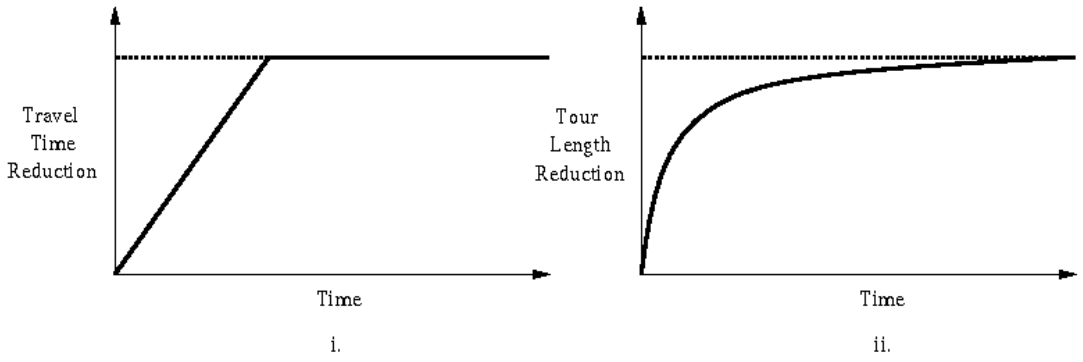

Figure 5: Performance profiles relating (i) the expected savings in travel time to time spent in path planning (ii) the expected reduction in the length of a tour as a function of time spent in tour improvement

iteration of search, and $d_{i,j}$ is the Manhattan distance from l_i to l_j[1]. A typical performance profile is illustrated in Figure 5 (i).

Let $T_{i,j}(\delta_j)$ represent the expected travel time between l_i and l_j given deliberation time δ_j. The average travel time of a path with no planning is $T_{i,j}(0) = 3.17 d_{i,j}/v$, where v is the velocity of the robot. The expected travel time for a path after maximum useful

[1]The values presented here are from (Boddy 1991), and differ slightly from those in (Boddy & Dean 1989)

deliberation is $T_{i,j}(\delta_j^*) = 1.17d_{i,j}/v$, and the expected travel time saved, $\mu(\delta_j)$, for $\delta_j \leq \delta_j^*$, is

$$\begin{aligned}
\mu(\delta) &= T_{i,j} - T_{i,j}(\delta) \\
&= \frac{3.17d_{i,j}}{v} - \frac{\delta}{\delta^*}\frac{3.17d_{i,j} - 1.17d_{i,j}}{v} \\
&= \frac{3.17d_{i,j}}{v} - \frac{1.50\delta/\tau_{pp}}{v} \\
&= \frac{3.17d_{i,j}}{v} - \frac{\gamma\delta}{v}
\end{aligned} \tag{11}$$

and $\mu(\delta) = 1.17d_{i,j}/v$ otherwise. We use γ to denote the *gain* of the algorithm.

3.3 Deliberation Scheduling Algorithms

Boddy and Dean present deliberation scheduling algorithms for three restricted versions of the robot-courier problem:

Pr-I Path-planning only, taking the tour as given

Pr-II Tour-improvement followed by path-planning

Pr-III A limited interleaving of tour-improvement and path-planning

Boddy and Dean provide optimal deliberation schedulers for each of these models, and compare their performance on a set of static problems.

3.3.1 Pr-I: Path-planning only

The first decision model that Boddy and Dean consider assumes that the robot can only deliberate about path-planning. Suppose the robot is given a tour $\mathcal{S} = \langle l_0, \ldots, l_k \rangle$, where l_0 is the robot's current position. Let δ_i be the time allocated to deliberating about the path from l_{i-1} to l_i, and let l_i.`begin` and l_i.`end` be the begin and end points, respectively, of the time when the robot is traveling from l_{i-1} to l_i. The robot must complete all deliberation about a path between two locations before it starts to travel between the two locations, the robot can deliberate about future paths as it moves, and the robot may stop and deliberate at any time.

Boddy and Dean (Boddy & Dean 1994) consider a few special instances of this problem. First consider a model where all the deliberations are performed before carrying out any

task. Deliberation scheduling is then simple: for each task allocate time as long as the gain of the anytime algorithm is greater than one. Now consider a problem similar to robot-courier, but where the tasks to be completed are not the same, and the performance profile may differ for each task. Boddy and Dean are not able to present a general solution to the problem, but some insights can be gained into the robot-courier problem. An initial step is to allocate a minimal allocation of deliberation to each task, the allocation that will result in the greatest reduction in the total time spent in execution. The deliberation is scheduled to occur during execution as much as possible. At the end of this stage there may be *free-time*, time when the robot is executing, but not deliberating; and *dead-time*, time when the robot is deliberating, but not executing. The goal of the algorithm is then to shift as much dead-time processing to free-time. This is illustrated in Figure 6, where t represents execution time, d dead-time, and f free-time. The unlabeled sections represent deliberation time happening together with task execution.

The procedure `(int Sched-1)` in Figure 7 generates the optimal deliberation schedule for the robot[2]. The procedure allocates deliberation times δ_i, $0 < i \leq k$, and returns the total length of the optimal tour, including deliberation time and journey time.

The correctness of the algorithm is proved by Boddy (Boddy 1991). The correctness is based on three properties:

1. Any optimal deliberation schedule can be transformed without cost into a deliberation schedule in which the robot deliberates about each path in turn, in the order in which they occur.

2. Any optimal deliberation schedule can be transformed without cost into a deliberation schedule in which the robot plans without moving only at the very start of the tour.

3. Any time allocated to deliberation δ_i could be allocated to δ_j with no increase in cost, so long as $\delta_i < \delta_i^*$ and $\delta_j < \delta_j^*$.

An example deliberation schedule is illustrated in Figure 8. In this case, $d_{0,1} = d_{1,2} = d_{2,3} = 100$, $\tau pp = 1$, and $\gamma > 1$. Note that if $\gamma < 1$ we never do any deliberation for $T_{0,1}$, because deliberation time is larger than the expected decrease in travel time. Even when $\gamma < 1$ however, we should deliberate during the execution of tasks, since this deliberation

[2]This algorithm is from (Boddy 1991) and differs from (Boddy & Dean 1989)

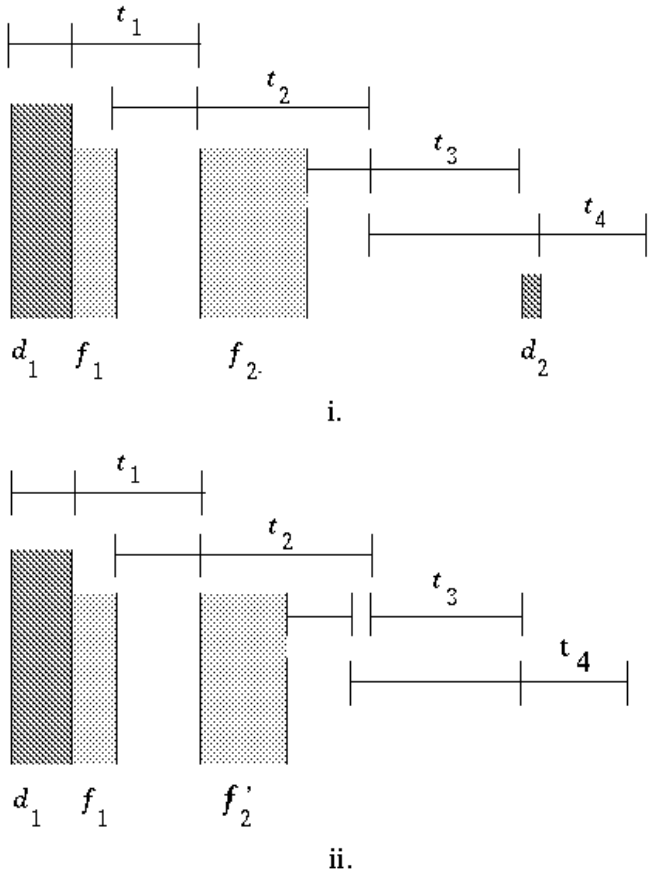

Figure 6: Reducing dead time using available free time

is essentially "free". The optimal solution is $\delta_1 = 112$, and $\delta_2 = \delta_3 = 133$, which gives $T_{0,1} = 149$, and $T_{1,2} = T_{2,3} = 117$. The algorithm first assigns $\delta_3 = \delta_3^* = 133$, since this deliberation can be done while traveling from l_0 to l_2, which will take at least 234 time units. Then the scheduling algorithm assigns $\delta_2 = \delta_2^* = 133$, since this deliberation can be done while traveling from l_0 to l_1, along with $(133 - 117) = 16$ units of deliberation for δ_3, so long as the travel time for $l_{0,1}$, $T_{0,1}(\delta_1) \geq 133 + 16 = 149$. The optimal solution is then found by allocating just the right amount of deliberation time to δ_1 so that there is no free-time while traveling from l_0 to l_1. Hence $\delta_1 = 112$ and $T_{0,1}(\delta_1) = 149$. Notice that if there was free-time while traveling from l_0 to l_1, then we could reduce the time for the overall solution by deliberating more, while $\delta_1 < \delta_1^*$, since $\gamma > 1$. Also notice that it is not optimal to make $\delta_1 > 112$, since this will cause dead-time in the schedule while the robot is at l_2, as it completes its deliberation δ_3. The total time (deliberation + travel) for this

```
int Sched-1(S) {
  t = l[k].begin();
  j = k;
  while ( (j != 0) && (t != l[1].begin()) ) {
    if (t > l[j].begin()) {
      gap = min(gamma * delta_star[j], t - l[j].begin());
      t = l[j].begin();
    }
    else
      gap = 0;
    if (t != l[1].begin()) {
(*)    delta[j] = min(delta_star[j], t - l[1].begin(),
                      (t -l[1].begin() + gap)/(gamma + 1) );
      t = t - delta[j] - max(0, gamma * delta[j] - gap);
    }
    j--;
  }
  if ( (j != 0) && (gamma > 1) && (gap > 0) ) {
    Delta = min( delta_star[j] - delta[j], gap / gamma);
    delta[j] += Delta;
  }
  for (i = 1; i < j; i ++) delta[i] = 0;
  for (i = 1; i <= n; i++) total += delta[i];
  return Delta + total;
}
```

Figure 7: Deliberation Scheduling

optimum schedule is $\delta_1 + T_{0,1} + T_{1,2} + T_{2,3}$.

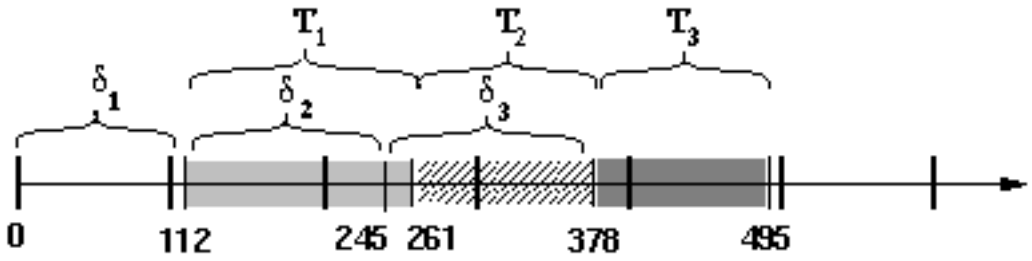

Figure 8: Optimal allocation for a simple deliberation problem

The procedure works backwards from the last path to be traversed, allocating deliberation time as it goes. The variable `gap` in the algorithm maintains the length of the earliest predicted free-time, that is travel-time with no worthwhile deliberation. The assignment of δ_j in line (*) tries to close the `gap` by setting deliberation to no larger than: the maximum useful time δ_j^*, the total time available $t - l_1.\texttt{begin}$, and $(t - l_1.\texttt{begin} + \texttt{gap})/(\gamma + 1)$. We must be allocating deliberation time for some path $j > 1$, and therefore any deliberation up to the start time of l_1 is useful. The third condition is derived by setting the execution time saved by deliberation so that the total time spent in execution is equal to that needed for deliberation. The problem can also be cast as a linear optimization problem (Boddy 1991).

3.3.2 Pr-II: Tour-improvement followed by Path-planning

We now allow the tour $\mathcal{S}$ to be reordered, but only before any path-planning. Given a tour $\mathcal{S}$ we can generate a schedule for path-planning using Pr-I. In order to reason about the value of tour-improvement we need to be able to reason about the expected time required (including deliberation) to complete a revised tour $\mathcal{S}'$. Remember that the tour-improvement algorithm does not know about the obstacles in the world, and that we must use the *expected* performance of path-planning to guide the allocation of reasoning to the algorithm. Boddy and Dean simplify the composition problem by assuming that the paths in tour $\mathcal{S}'$ are all of equal length, $d = \mathcal{D}_{\mathcal{S}'}/k$, where there are k locations to visit, and $\mathcal{D}_{\mathcal{S}'}$ is the total length of tour $\mathcal{S}'$. Boddy and Dean develop a linear expression for the expected time to complete a tour, given the average path length d, written $f(d) = \alpha d$. The performance of the tour-improvement algorithm is represented with the function $\hat{\mathcal{D}}_{\mathcal{S}'} = g(\mathcal{S}, \delta)$, where δ is the deliberation time assigned to tour-improvement. The optimal deliberation schedule

allocates deliberation time δ to solve:

$$\delta^* = \arg \min_{\delta} [\delta + (\alpha g(\mathcal{S}, \delta)/k)] \quad \text{s.t.} \quad \delta \geq 0 \tag{12}$$

The minimization can be done easily because the performance profile of the path-planning algorithm is smooth, convex, and monotonically increasing.

3.3.3 Pr-III: A limited interleaving of the two algorithms

The general problem of interleaving is hard, but Boddy and Dean present a polynomial-time algorithm for a restricted case. Boddy and Dean allow a limited interleaving of the two anytime algorithms: the robot may plan paths for locations $\langle l_0, \ldots, l_i \rangle$ using Pr-I, and then solve for locations $\{l_i, \ldots, l_k\}$ using Pr-II. The algorithm works by assuming initially that all the travel time for the locations $\langle l_0, \ldots, l_i \rangle$ will be used to deliberate about finding an improved tour for $\{l_i, \ldots, l_k\}$. The algorithm then updates this schedule to include deliberation for path-planning, where it has a greater expected value than tour-improvement, and seeks an optimal value for the partition $\{0, \ldots, i\}\{i, \ldots, k\}$.

3.3.4 A comparison: Pr-I, Pr-II and Pr-III

Boddy (Boddy 1991) performed a series of experiments involving randomly-generated environments and sets of locations for the robot to visit. In both static and dynamic environments (where the list of requests for service arrives over time), Pr-II is a substantial improvement over Pr-I, but Pr-III has less of an advantage over Pr-II, particularly in the dynamic case. The results for the static case are illustrated in Figure 9. The figure plots expected time for tour-completion against the size of the tour. Clearly, Pr-II and Pr-III are better than Pr-I, especially as the tour size increases and the adavantage or tour-improvment grows. The difference in performance between Pr-II and Pr-III is too small to see.

The algorithm presented for the robot-courier problem is optimal for a special case in which all of the PPs are piecewise linear composed of two linear segments such that the slope of the first segment is the same for all profiles and the slope of the second is 0.

3.4 Deliberation Scheduling in response to events in the world

Dean and Boddy are able to present an optimal solution to the simpler problem of responding to events in the world (Boddy & Dean 1994). The agent knows about some set

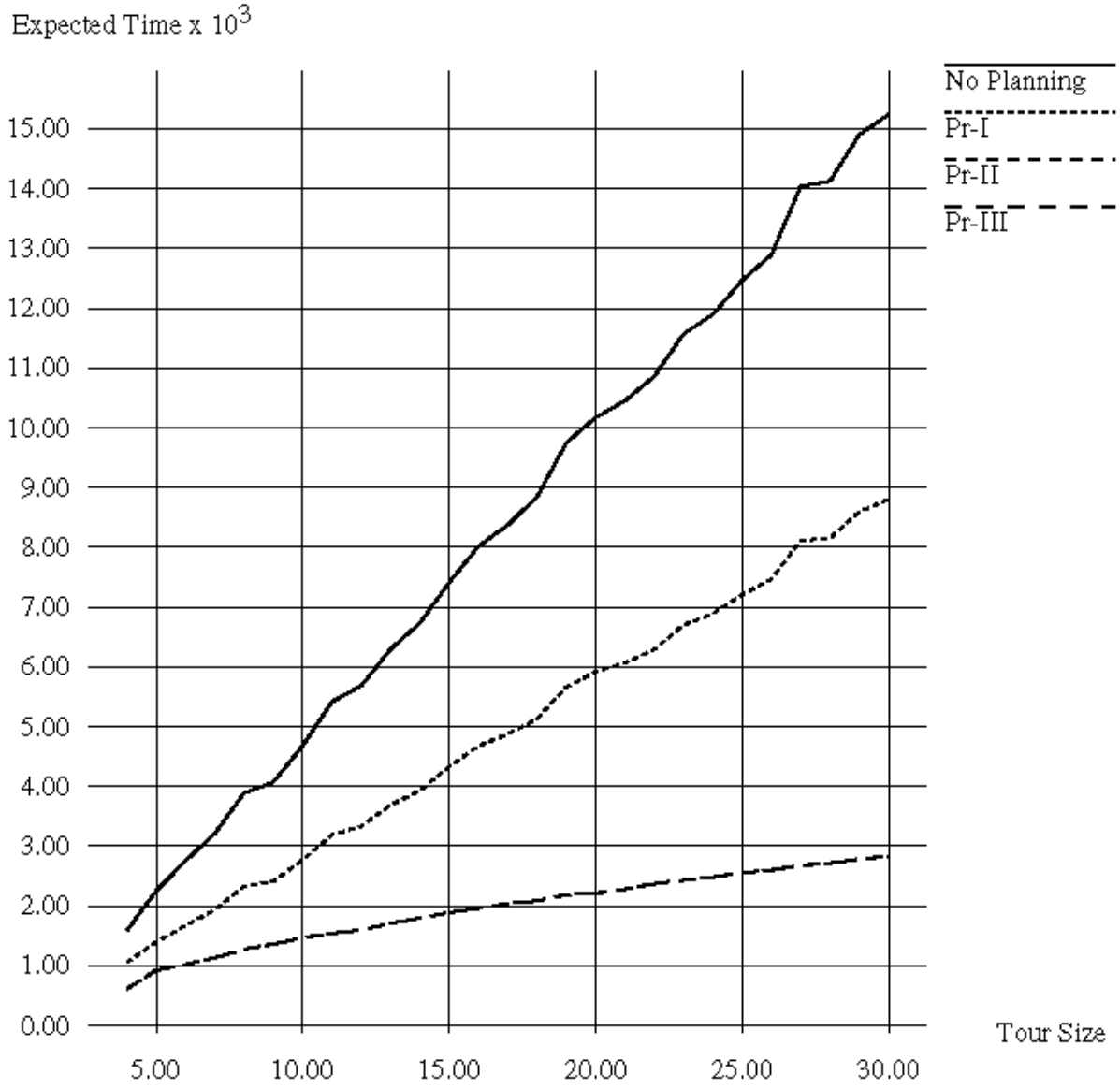

Figure 9: Varying tour size

of pending events that it has to formulate a response to. This problem is simpler because the decision processes are *independent*. In the robot-courier problem the decision processes are actually *dependent* because a decision to spend more time reasoning on one problem reduces the time available for reasoning on a later problem by reducing the execution time during which deliberation is possible. Assuming a diminishing returns performance profile Boddy and Dean are able construct optimal deliberation-schedules using a polynomial-time algorithm (Boddy & Dean 1994). Boddy and Dean extend the algorithms to the case where

the exact time of the occurrence of conditions is not known.

The agent constructs a schedule that allocates some amount of time to some of the decision procedures, not allocating any time after the even occurs. Dean and Boddy describe a pair of algorithms for finding optimal schedules (Boddy & Dean 1994). The first algorithm (DS-1) assumes that the gain for a performance profile at any particular time is available. The algorithm schedules backwards from the latest deadline to the current time, in fixed increments of size Δ. It simply assigns each increment to the executable tasks with the largest gain. After each Δ has been assigned the scheduler goes back and gives each task a contiguous block of time. They show for any $\epsilon > 0$ there is a $\Delta > 0$ such that DS-1 constructs a schedule within ϵ of optimal. The second algorithm, DS-2, assumes that the performance profiles can be described in terms of known functions. This algorithm also schedules backwards from the last deadline, and time is allocated to each task according to a set of linear equations. The exact structure of the equations depends on the form of the performance profile functions.

3.5 Boddy and Dean: Summary

Boddy and Dean study the problem of metareasoning for a specific class of decision processes. Their work on reasoning about anytime algorithms can be viewed as a refinement of Russell and Wefald's general framework for metareasoning. Boddy and Dean do not consider metareasoning for a single problem, indeed that would be trivial for an anytime algorithm with a known performance profile. Metareasoning for anytime algorithms is termed "deliberation scheduling", and the problem is cast as an optimization problem: what is the allocation of resources to a set of anytime algorithms that maximizes the expected utility of the agent? Boddy and Dean consider the problem of an agent responding to a sequence of events in the world, and present an efficient algorithm for deliberation scheduling that optimizes the response of the agent, given a set of anytime algorithms and their associated performance profiles. Boddy and Dean ignore the cost of metareasoning, and are essentially doing reasoning at the meta-level off-line.

The robot-courier case study is an interesting analysis of the complexity of scheduling optimal reasoning for a resource-bounded agent acting in a real-time environment. However, the algorithms for deliberation scheduling presented are very specific, and make strong assumptions. In particular, the scheduling algorithm Pr-I only produces an optimal sched-

ule when the anytime algorithms for each task have *identical* performance profiles. This is unreasonable for an agent acting in any interesting domain, where the agent must heterogeneous tasks, and could have a choice of anytime algorithms to reason about any one task.

Boddy and Dean model deliberation in terms of continuously interruptible procedures, so metareasoning involves what to run, when, and for how long. Boddy and Dean emphasize events and their temporal relationships, something which is difficult to do within Russell and Wefald's framework, where the only explicit consideration of time is in the time cost of reasoning. Boddy and Dean motivate the composition problem for anytime algorithms, that is how to schedule deliberation to a system of dependent anytime algorithms , which is the main thesis of Zilberstein's (Zilberstein & Russell 1996) later work.

4 Horvitz: Flexible Computations

Horvitz (Horvitz 1987) takes a similar approach to that of Boddy and Dean. He also proposes anytime algorithms, he terms them *flexible computations*, for decision processes. The main focus of his work is on the control of probabilistic inference in medical decision-making. There are parallels to the work of Russell and Wefald, in that Horvitz emphasizes the need to apply normative (decision-theoretic) reasoning at the meta-level when insufficient resources prohibit a complete normative analysis at the base-level. The goal is to extend the principles of normative rationality to situations of uncertain, varying and scarce resources.

Traditional decision theory has the desirable property that it finds provably optimal solutions to problems. Unfortunately the updating of beliefs using probabilistic inference is $\mathcal{NP}$-hard (Cooper 1990), making traditional decision-theory unsuitable for most real-time situations. Horvitz seeks to apply normative methods to provide bounded-rational solutions to otherwise intractable problems.

Rational reasoning at the meta-level necessarily requires knowledge about the inference costs, and real-world costs and benefits. Horvitz terms the expected utility of computation, the *comprehensive value*, V_c. This is composed of the *object-related* value, V_o, and the *inference-related* value, V_i. The object-related value is the expected utility associated with the best action available to an agent, given a reference state of the world, and is exactly Russell and Wefald's intrinsic value. The inference-related value is the expected disutility

of computation, due to changes in the world. The *net value of computation*, ΔV_c, refers to the change in the comprehensive value, given some computation.

Horvitz uses vectors of attributes to define a computational state. The current state is written $\langle \vec{v}, \vec{r} \rangle$ where $\vec{v}$ and $\vec{r}$ represent object-level and inference-level attributes, respectively. Since there is always uncertainty in the object-level state that results from a computation, we must necessarily talk about the *expected* net value of computation:

$$\Delta V_c = \sum_{\vec{v}'} V_c(\vec{v}', \vec{r}') \Pr(\vec{v}' | \vec{r}') - V_c(\vec{v}, \vec{r}) \tag{13}$$

where $\vec{v}$ and $\vec{r}$ are the attributes before computation, $\vec{v}'$ and $\vec{r}'$ are the revised attributes, and $\Pr(\vec{v}'|\vec{r}')$ is the probability of the object level attributes $\vec{v}'$ given inference level attributes $\vec{r}'$. This is similar to equation 5 of Russell and Wefald, a computation has value if it is expected to lead to a better base-level action. Within this framework, Horvitz proposes using knowledge of the *value structure* of timewise-refinement algorithms to make the appropriate cost-benefit tradeoffs (performance profiles). Horvitz lists some desiderata of decision-processes when reasoning under scarce resources, the primary of which is *flexibility*. Flexible computations are robust to unexpected interruption, provide a tradeoff between output quality and computation time, and guarantee monotonically increasing quality.

Horvitz introduces the notion of *bounded optimality*, and the weaker concept of *bounded strategic optimality*. Bounded optimality refers to the optimization of computation utility given certain assumptions about the environment, and resource constraints. Strategic bounded optimality is the optimization of utility, given a set of strategies from which to choose. Boddy and Dean construct a bounded strategic optimal schedule for the robot-courier problem, the solution they generate is optimal *given* the two anytime algorithms available. Horvitz also proposes the *independent challenge model* as a means to compare the relative performance of different agents. The independent challenge model assesses the performance of an agent in response to a distribution of independent problems. Given a distribution of problems, P_i, over a period of time, t, where $f(P_i)$ is the frequency of problem type P_i, the optimal agent A^* should solve

$$A^*(t) = \arg \max_A \sum_{i=1}^{n} t f(P_i) \left(V_d(A, P_i) + \Delta V_c(S^*(A, P_i)) \right) \tag{14}$$

, where $V_d(A, P_i)$ is the expected value associated with agent A taking its best default action in response to a problem P_i, and $\Delta V_c(S^*(A, P_i))$ is the incremental value of the best computational strategy S available to agent A. The best agent has a set of computational reasoning strategies $S(A)$ that maximizes its utility over the distribution of problems that might occur. Within the independent challenge model we can reason about the relative performance of agents, the value of learning, and the utility of adding a new capability to an agent.

4.1 Reasoning about the Time-Precision Tradeoff

Horvitz presents a problem from medical decision making. An automated control system is faced with a rapidly evolving set of respiratory symptoms in a patient in an intensive-care unit. The pulmonary decision-making system must provide estimates of the probability, $p(C|E)$, of various conditions, C, given observed symptoms, E, so that the physician is able to take the best action.

Horvitz uses a probabilistic bounding strategy for the computation, that determines upper and lower *bounds* on point probabilities of interest through a logical analysis of constraints acquired from partial analysis. Bounds become tighter as additional constraints are brought into consideration (Cooper 1984). The precision of this decision procedure, S_1, increases with time as illustrated in Figure 10 (a).

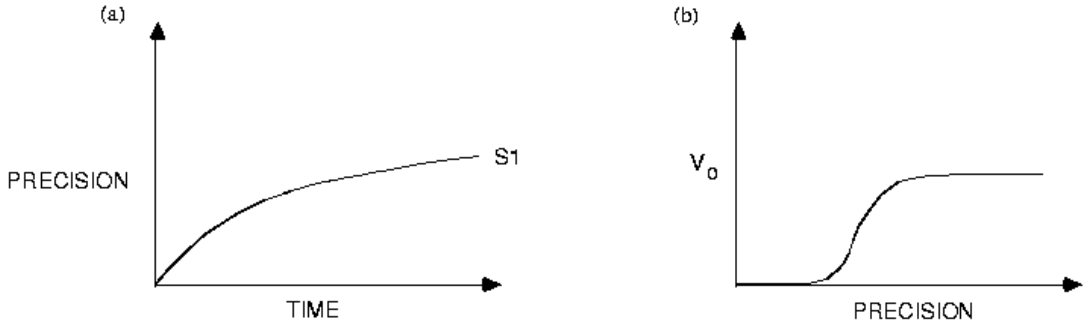

Figure 10: (a) The timewise refinement of the precision of a result with additional computation. (b) The object-level utility of the refined result.

The physician then provides the expert knowledge to enable values to be assigned to partial results. This is done on the basis of a cost analysis of the four possible outcomes of the prediction: { `true-positive, true-negative, false-positive, false-negative`

} . Horvitz is then able to create a performance profile for the decision-process using this information. This is illustrated in Figure 10 (b).

A traditional AI system, using normative reasoning and considering only intrinsic-value, would aim to maximize the intrinsic value of the decision, by computing the reasoning until its decision is within pre-specified accuracies, by which time the patient could be in a critical condition. The emphasis here, of course, is on considering the cost of deliberation explicitly within the reasoning process. The *comprehensive value* of the result decreases with the amount of time that the computation takes, and in this case the cost of deliberation to the patient is clear. Horvitz assumes that the cost of delay can be represented as an independent multiplicative discounting factor, D_t, ranging between one and zero. The curve is illustrated in Figure 11 (a).

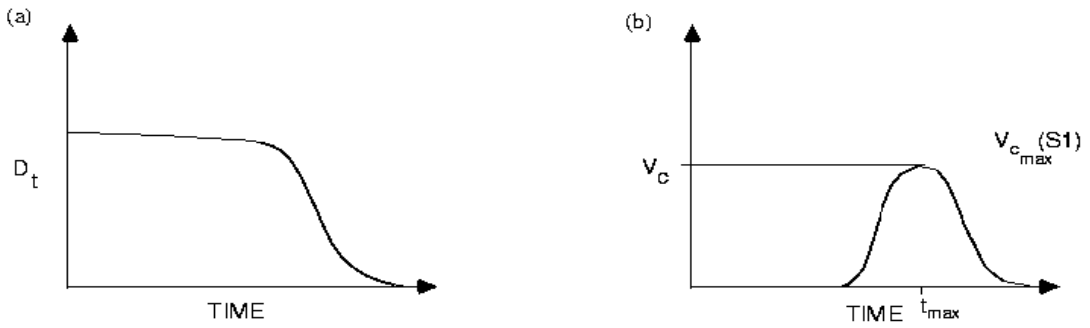

Figure 11: (a) The inference-related cost based in delay of action. (b) The comprehensive value of computation.

We combine the three curves to attain a profile of comprehensive value to the physician, and it is this value that a rational pulmonary decision-making system should optimize. Horvitz is able to present a clear analysis of the cost of time in a time-critical domain. The result is illustrated in Figure 11 (b).

Notice that the comprehensive value has a global maximum $V_{c_{\max}}$ at a particular time, $t_{\max}$. This is the period of time the computer should apply the inference scheme, before making a recommendation to the physician. Additional time will increase precision, but decrease the comprehensive value of the reasoning due to costs of delaying the decision.

4.2 Multiple Decision Procedures

The framework also allows Horvitz to present a qualitative analysis of the choice between alternative reasoning strategies, given their performance profiles. Suppose we have another decision procedure S_2. Given a particular time constraint we might choose either S_1 or S_2. This is illustrated in Figure 12.

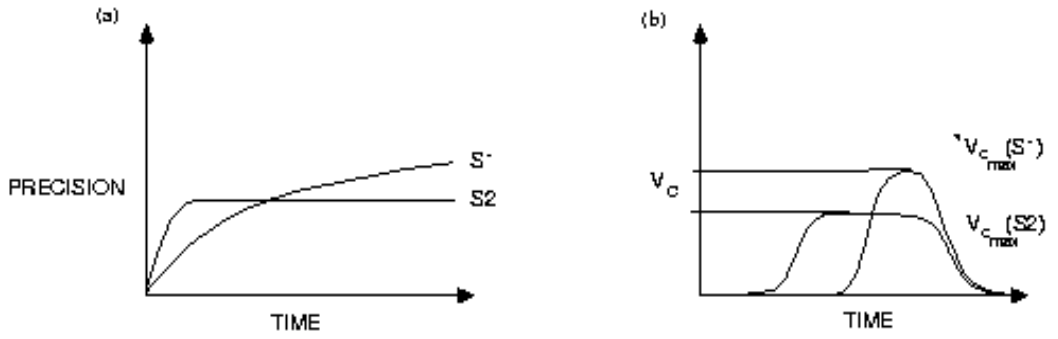

Figure 12: (a) The timewise refinement of another inference strategy. (b) A comparison of the comprehensive value of computation of the two inference strategies.

A default rule for a particular context can be made with little computation, and has a object-related value that is constant with time since it is not refined. In situations of extreme time criticality we can expect a default (compiled) policy with low objective-value to be the best strategy. This tradeoff is illustrated in Figure 13.

4.3 A general framework for metareasoning

Horvitz has also looked at the general problem of partitioning resources between metareasoning and base-level problem solving (Horvitz & Breese 1990). In particular, he examines how much time to devote to deliberation-scheduling when both scheduling *and* base-level reasoning are carried out using anytime algorithms. For particular classes of anytime algorithms, Horvitz and Breese are able to find optimal solutions to the problem of metareasoning. They also extend their analysis to incorporate decisions about learning and compilation, which bring *long-term* benefits, at the meta-level. In all the cases the costs of the *meta-meta-* level are a small constant because they involve plugging particular values into a mathematical function. Russell and Wefald investigate similar problems *without* assuming the availability of anytime algorithms in either the metareasoning or base-level computations. They avoid the meta-meta problem by making the meta-greedy assumption.

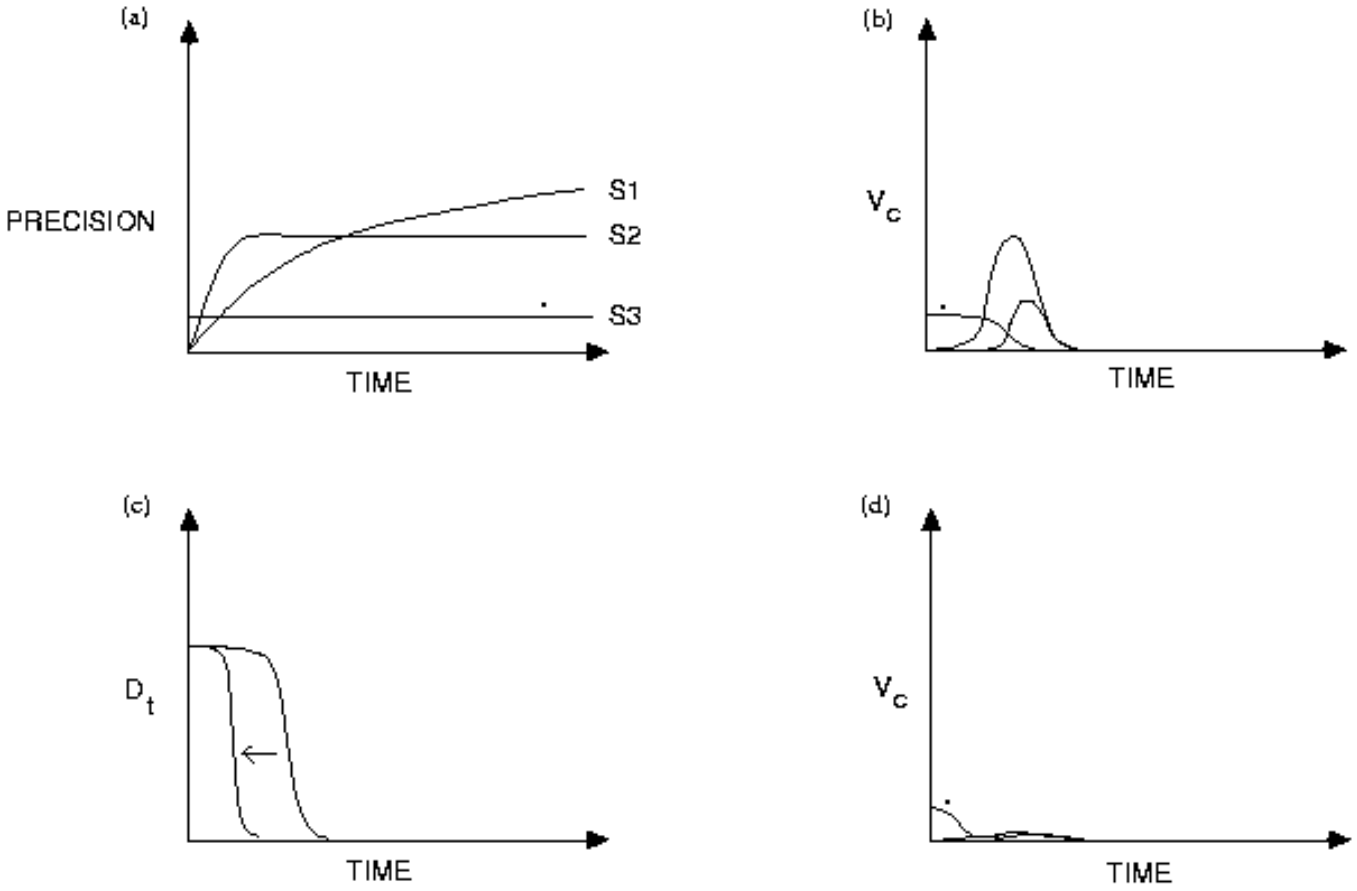

Figure 13: (a) A default reasoning strategy (marked). (b) The comprehensive value of the default strategy. (c) Another shift in the decision horizon. (d) The dominance of the utility of the default strategy in a more critical decision-making context.

4.4 Horvitz: Summary

Horvitz developed an approach to bounded-rationality using flexible computations, independently of Boddy and Dean. He notes that flexible computations are useful for bounded-rational agents that must do metareasoning because they: are robust to unexpected interruption; provide a tradeoff between output quality and computation time; and guarantee monotonically increasing quality. The approach is, like that of Boddy and Dean, essentially off-line compilation of optimal control of reasoning, in this case for a single decision problem. Horvitz does provides a more general analysis of the time cost of deliberation than that of Boddy and Dean, who either assume that a response must happen at a certain time, or that the goal is to minimize total action and deliberation time, in which case the

cost of deliberation is simply additive. Horvitz gives a qualitative analysis of the tradeoff between quality and time-cost in the choice between multiple decision procedures for the same problem. Horvitz presents an example of metareasoning for a decision-process from the medical decision-making domain, and presents an analysis of how the comprehensive value of the anytime algorithm is developed. The main contributions of his work are:

- determining how to assign utility values to partial results
- finding anytime algorithms to implement reasoning
- defining the comprehensive value of a computation, and reasoning about its object-related and inference-related components
- solving single real-time problems with flexible computations

Horvitz also originated the term *bounded optimality*, and discussed the problems of traditional normative reasoning for real-time AI.

5 Zilberstein: The Composition of Anytime Algorithms

Zilberstein and Russell (Russell & Wefald 1991; Zilberstein 1993; Zilberstein & Russell 1996) focus on the *composition problem*, that is how to schedule deliberation to a system of dependent anytime algorithms. The two basic problems addressed are the *interruptibility* of the composed system, and how to allocate time optimally among components. The goal is to allocate time to each algorithm so as to maximize the expected utility of the system as a whole. Zilberstein and Russell present efficient algorithms for a restricted class of modular anytime systems. The solutions are optimal if each of the performance profiles is a monotonic non-decreasing function of input quality. Zilberstein and Russell also describe heuristic algorithms for the general problem, which is shown to be $\mathcal{NP}$-complete. Zilberstein introduces the important concept of a *conditional* performance profile, that maps the quality of the output of the algorithm as a function of time, *given a certain input quality.*

Zilberstein and Russell construct *contract* anytime algorithms. A contract anytime algorithm returns increased value as it is given more time, but must be told in advance how much time it is going to get, and may not return any answer if it is given less time than prescribed. Constructing contract anytime algorithms using compilation of anytime modules is easier that constructing interruptible anytime algorithms directly. Intuitively,

the design is easier because it is not necessary to provide for a solution at any time other than the specified computation time. We can produce an optimal allocation of computation from the contract time to each module, without concern for interleaving the computations to provide partial results at any time. The result of the compilation is a schedule that specifies the amount of run-time for each component anytime algorithm.

Zilberstein and Russell prove that an interruptible algorithm can be constructed from any contract algorithm at some small fixed cost. Zilberstein provides a construction that transforms any contract algorithm to an interruptible algorithm.

Theorem (Reduction). For any contract algorithm, $\mathcal{A}$, an interruptible algorithm $\mathcal{B}$ can be constructed such that for any particular input $q_{\mathcal{B}}(4t) \geq q_{\mathcal{A}}(t)$.

The proof is constructive. We simply run $\mathcal{A}$ repeatedly with exponentially increasing time limits. If interrupted, return the best result generated so far. In the worst case the computation is interrupted, at time t, just before the most recent call to $\mathcal{A}$ terminates, and the quality will be at least that of $q_{\mathcal{A}}(t/4)$. In the best case we have a performance that is only twice as bad as that of the contract algorithm, run for the same time. The contract algorithm can achieve in time t what might take up to $4t$ in the interruptible algorithm.

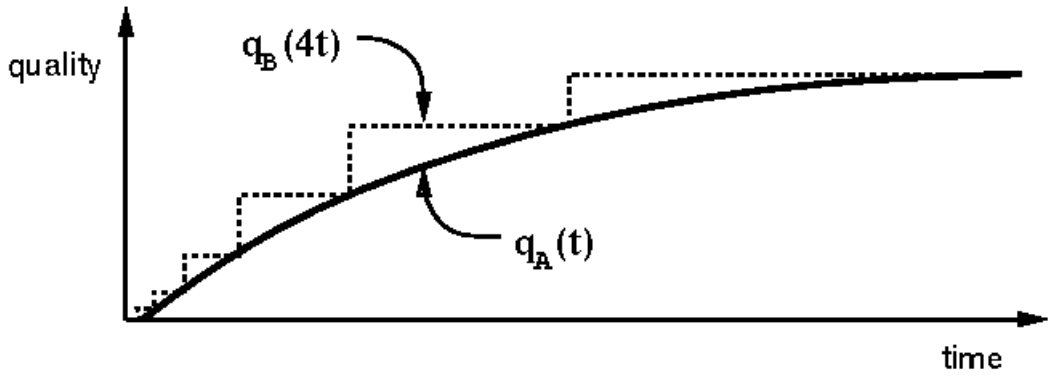

Figure 14: Performance profiles of interruptible and contract algorithms

5.1 Example: A simulation method to acquire a performance profile

Zilberstein and Russell present an application of an anytime algorithm for the classic TSP problem, that is known to be $\mathcal{NP}$-complete. The iterative improvement algorithm repeatedly tries to exchange r edges in a feasible tour for r edges not in the solution, as long as

the tour remains feasible, and the cost is strictly less. Zilberstein illustrates the *quality map* for the algorithm, where each point (t, q) represents an instance for which quality q was achieved with run-time t. The quality measures the percentage of tour length reduction with respect to the initial tour. The statistics form the basis of the expected performance profile for the algorithm. See Figure 15.

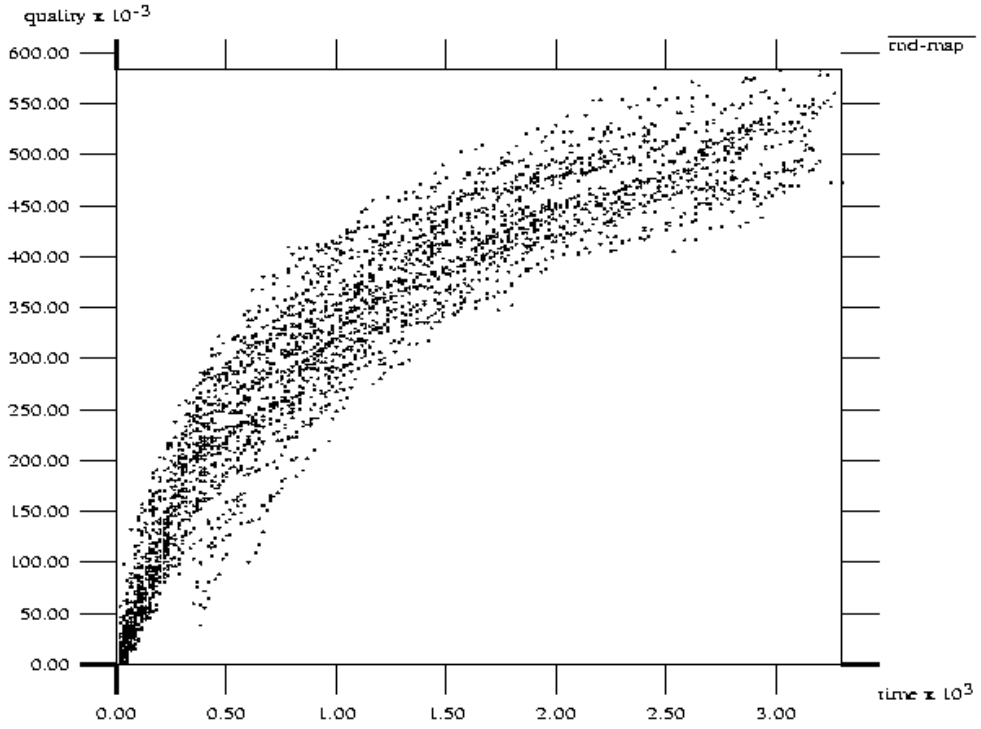

Figure 15: The quality map of the TSP algorithm

From the quality map of the performance of an algorithm $\mathcal{A}$ we can generate the expected performance profile (EPP), a function $E_{\mathcal{A}} : \Re^+ \rightarrow \Re$ that maps computation time to the *expected* quality of results (this is the performance profile used by Boddy and Dean (Boddy 1991)).

Zilberstein and Russell introduce three new types of performance profiles, that allow a trade-off between accuracy, compactness of representation, and efficiency for reasoning. The first is the *performance distribution profile* (PDP). The PDP of an algorithm $\mathcal{A}$ is a function $D_{\mathcal{A}} : \Re^+ \rightarrow \Pr(\Re)$ that maps computation time to a probability distribution over results. The performance interval profile (PIP) of an algorithm $\mathcal{A}$ is a function $I_{\mathcal{A}} : \Re^+ \rightarrow \Re \times \Re$ that maps computation time to upper and lower bounds of the quality of results. Russell and

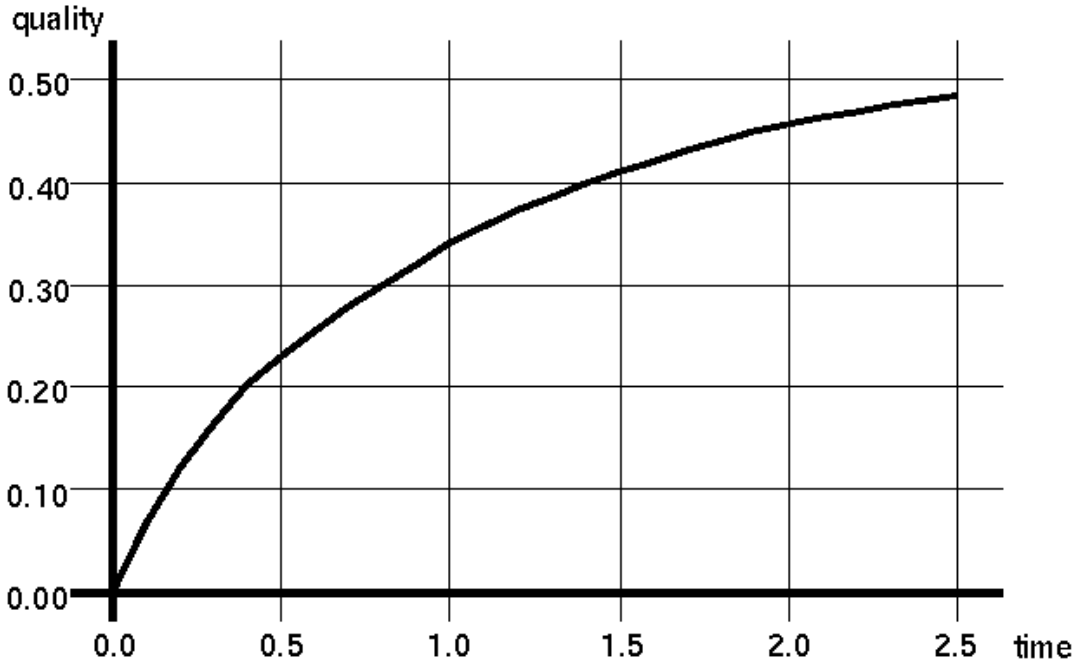

Figure 16: The expected performance profile of the TSP algorithm

Zilberstein propose a *conditional performance profile* (CPP), that is a mapping from input quality and run-time to probability distribution of output quality, $C_{\mathcal{A}} : \Re \times \Re^{+} \rightarrow \Pr(\Re)$. This is useful when composing modules whose performance depends significantly on input quality. The input is classified according to a vector of features, and partitioned into classes, with a separate performance profile representing each input class. The CPP allows rational reasoning about the best allocation of resources to a set of anytime algorithms. A typical CPP is illustrated in Figure 17.

5.2 Compilation of anytime algorithms

Zilberstein suggests compilation as a solution to the composition problem. The compiler takes the anytime modules and performance profiles as input, and generates a deliberation schedule for the overall system, along with a monitor, and a performance profile. A *monitor* controls the run-time execution of the anytime algorithm. It is initially assumed to be passive, but the idea generalizes to active monitors that monitor the progress of the quality of the output from the anytime algorithms within the system, and allow for adaptive control of reasoning.

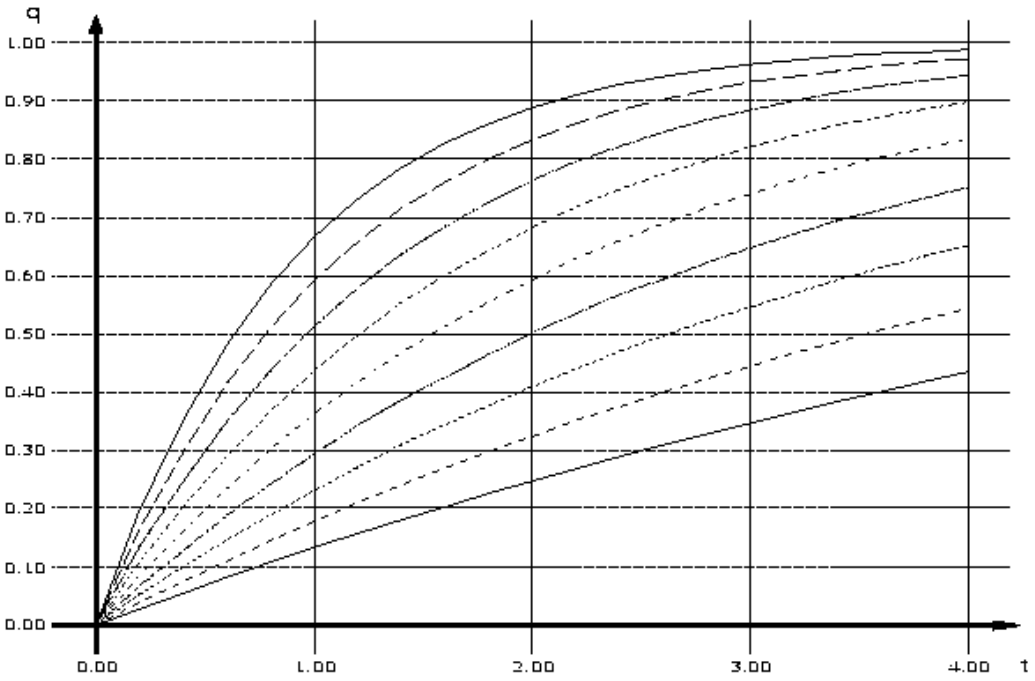

Figure 17: Graphical representation of a conditional performance profile

5.3 Example: Composition of sensing and path-planning

Zilberstein and Russell consider a robot navigation system that is composed of two anytime algorithms: visual sensing and path-planning. The output quality of path-planning depends on the input quality from sensing, and this dependence is represented using a conditional performance profile (Figure 18). Performance profiles are represented as discrete tables, and the compilation of the two modules is then cast as a discrete optimization problem. The compiled algorithm is shown to be superior to two heuristic deliberation schedules. If a discrete tabular representation is used the compilation problem grows exponentially in the number of modules.

5.4 Compilation for the general composition problem

Zilberstein present heuristics for the general case of functional composition of anytime algorithms. Let $\mathcal{F}$ be a set of anytime functions, and assume that each function $f \in \mathcal{F}$ has a fixed conditional performance profile. In general the complexity of the compilation

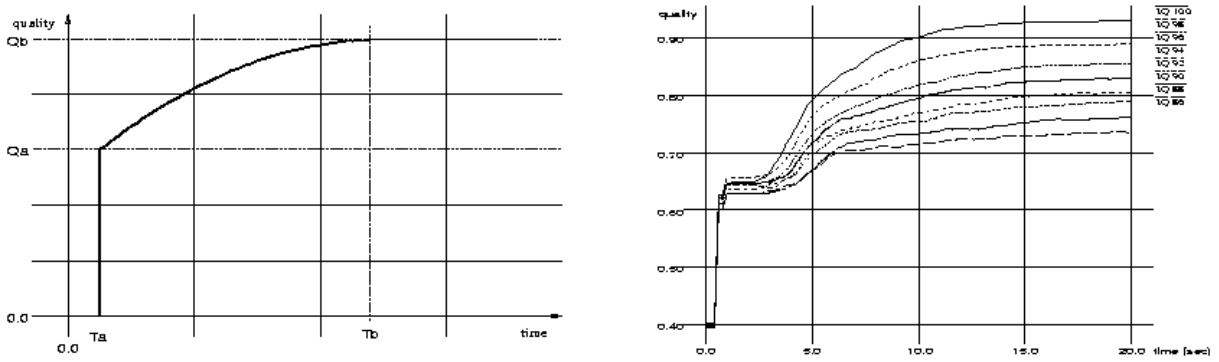

Figure 18: (a) Expected performance profile of visual sensing (b) Conditional performance profile of path-planning, given an input quality between 0.86 and 1.0

problem depends on the structure of the composition. Consider

$$F(x) = E(D(B(A(x)), C(A(x))))$$

Two possible representations are shown in Figure 19.

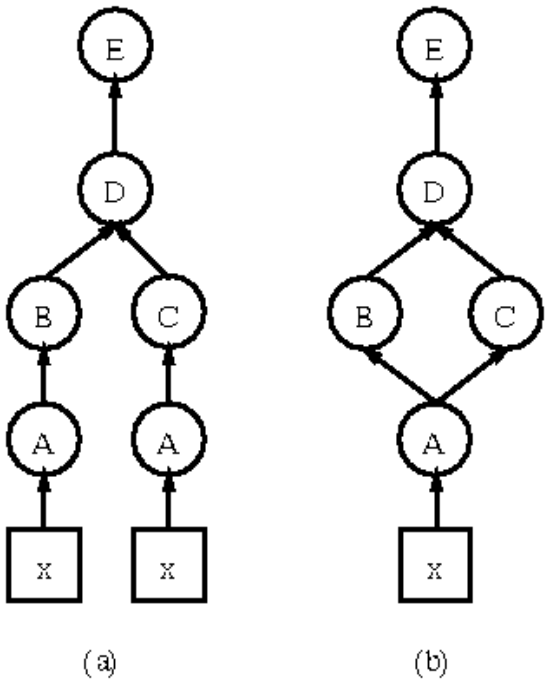

Figure 19: Graph representation of functional expressions

When the dependencies are unrestricted, and form a directed acyclic graph, the problem

is $\mathcal{NP}$-complete. Given a composite expression, the conditional PPs of its components, and a total time allocation B, the composition problem is expressed as a decision problem: Is there a schedule of the components that yields output quality greater than or equal to K? The general problem is shown to be $\mathcal{NP}$-hard by reduction from the `PARTIALLY ORDERED KNAPSACK` problem (Zilberstein 1993). The meaning of this result is that the application of the compilation technique is limited to small programs. The tree-structured problem with no repeated sub-expressions is also shown to be $\mathcal{NP}$-complete, by reduction from the `KNAPSACK` problem. There is, however, a glimmer of hope. The `KNAPSACK` problem is *pseudo-polynomial*, that is polynomial given bounded input sizes, and can be efficiently solved using a dynamic programming algorithm for that case.

The same technique can be applied to scheduling in a tree-structure composition problem. We use *local compilation* to optimize the quality of output of each elementary anytime algorithm by considering only the PPs of its immediate subcomponents (without regard to its wider context). Local compilation replaces the global optimization problem with a set of simpler, local optimization problems, and reduces the complexity of the whole problem. The tree-structured assumption is needed so that local compilation can be applied (without it we have infinite recursions). If a subcomponent is not elementary then its performance profile is determined by local compilation as well. Whenever we have input monotonicity local compilation is shown to be globally optimal. Finally, when the number of inputs to each function is bounded we have a pseudo-polynomial algorithm for the global problem. This assumption is consistent with one of the tenets of good code design - modularity. With a compact tabular representation of PPs, local compilation can be performed in constant time, and the overall complexity of compilation becomes linear in the size of the program (Zilberstein & Russell 1996).

5.5 Heuristics for the general problem

Zilberstein present three heuristic time allocation methods that deal with general functional expressions: `HILL-CLIMBING-ALLOCATION, CONDITIONING-ALLOCATION, and TRADING-ALLOCATION`. The choice of algorithm to use represents a tradeoff between guaranteeing optimality, and achieving an efficient algorithm. Some techniques work efficiently on DAGs and produce near-optimal schedules, other techniques guarantee optimality when the number of repeated subexpressions is small.

5.6 Run-Time Monitoring

The basic approach to compilation can be viewed as off-line meta-reasoning, where a passive monitor is used to control the deliberation of the anytime algorithms exactly as computed off-line. The purpose of run-time monitoring is to reduce the effect of uncertainty on the performance of the system. Monitoring can vary in the frequency of monitoring, the approximation methods used to assess solution quality, and the degree to which the monitor is active. More generally, monitoring allows a trade-off between off-line compilation and run-time deliberation.

An active approach to monitoring is useful when the utility function is not predictable. Zilberstein (Zilberstein & Russell 1996) suggests using the expected marginal value of computation to continually make resource scheduling decisions. The monitor estimates the difference between the expected utility after one more time increment and the current expected utility. This myopic approach is optimal when the PP is monotonically increasing and concave down, and the cost of time is monotonically increasing and concave up. Zilberstein is applying normative control to the metareasoning problem for anytime algorithms, in the same way the Russell and Wefald apply normative control to search.

Monitoring might involve complex computation, and in this case we need a model of the interruptible anytime algorithm. A Markov model is used to represent reasoning, states are identified with particular solution qualities, and transitions between states correspond to improvement in solution quality as a result of a small increment in computation time. This approach allows one to calculate an off-line monitoring policy that determines an optimal action for each solution quality. Possible actions are: (1) resume the execution of the algorithm for a certain time interval and then monitor again, (2) resume the algorithm again for a certain time interval and then stop, or (3) stop the execution of the algorithm. A simple example for the Traveling Salesperson Problem is shown in Figure 20.

The table specifies, for any given time step and solution quality, the amount of additional time to be allocated to the algorithm. The letter M indicates that monitoring should be performed again after the time allocation. An interesting property is that they recommend monitoring more frequently near the expected stopping time of the algorithm (intuitive). When solution quality cannot be determined at run-time, a similar policy can be developed based on estimated solution quality (Hansen & Zilberstein 1996).

	time-step											
quality	start	1	2	3	4	5	6	7	8	9	10	11
5		0	0	0	0	0	0	0	0	0	0	0
4		1M	1M	1M	1M	1M	1M	1M	1M	1M	1	0
3		1M	1M	1M	1M	1M	1M	1M	1M	1M	1	0
2		3M	3M	3M	3M	3M	3M	3M	3M	2	1	0
1		4M	4M	4M	4M	4M	5	4	3	2	1	0
0	5M	5M	5M	5M	5M	6	5	4	3	2	1	0

Figure 20: Optimal policy based on actual solution quality

5.7 Zilberstein and Russell: Summary

Zilberstein and Russell are able to show that a system of modular anytime algorithms can be compiled efficiently into a global anytime system, given certain constraints on the form of the compositions. When there are no repeated sub-expressions we have an optimal algorithm that is also efficient, using branch-and-bound techniques. The compilation process produces an optimal contract algorithm, that can be made interruptible with only a small, constant penalty. Solving the composition problem is important because it allows modular system development, and solves the complex task of activation and interruption of components for the programmer.

Zilberstein and Russell go a long way towards an *operational* theory of rationality, using anytime algorithms as their model. Zilberstein is able to avoid, to some degree, the myopic assumptions at the meta-level made by Russell and Wefald, by using dynamic programming to generate a non-myopic stopping rule off-line. Active monitoring also allows base-level reasoning to be controlled using estimates of the marginal value of further deliberation. Their approach to metareasoning is a mix between the off-line metareasoning of Boddy and Dean (passive monitors), and the dynamic metareasoning (active monitors) of Russell and Wefald. Zilberstein and Russell allow an explicit trade-off between pre-compilation of metareasoning and active monitoring. Active monitoring is useful when the performance profile of an algorithm is uncertain, although it does reintroduce myopia to the meta-level decision level.

The future goal of Zilberstein's work is to develop a complete theory of anytime-

rationality for an agent situated in the world. Current research efforts are aimed at studying additional programming structures, producing a large library of anytime algorithms (Grass & Zilberstein 1996), and extending the model to include anytime sensing and anytime actions (Zilberstein 1993).

6 Conclusions

We have argued that the concept of intelligence for resource-bounded agents must necessarily be defined in terms of the reasoning that the agent does, given its resources, and a model of the environment. The appropriate concept of rationality, is bounded- optimality. This states that no agent with the same resources, and the same model of the environment, can achieve a higher expected utility in the world. We clearly need normative control of reasoning, that is the agent should reason about what to reason about next, and for how long. Russell calls this operational definition of rationality *meta-level rationality.* The work that we survey here is an attempt to provide normative control of reasoning for bounded-resource agents operating in time-critical environments. We reassert the thesis of the work surveyed here: the study of resource-bounded agents should be *central* to AI, since perfect rationality and epistemological correctness provide neither an adequate theoretical or practical framework.

Russell and Wefald (Russell & Wefald 1991) present a powerful framework for metalevel rationality, using information value theory (Howard 1966), to derive conditions that define how the agent should deliberate at the base-level. The "external model" that they promote underlies the theory of normative metareasoning. The idea is that we should take decisions at the metalevel on the basis of the value of deliberation, and that the value of deliberation is solely derived from any change in external action that further reasoning might lead to. Russell and Wefald control reasoning at the metalevel with myopic heuristics, and demonstrate, through an application to a competitive game-playing problem, the tremendous increase in rationality (as judged by utility in the world) that comes from principled metareasoning. The main limitations of their work are: the uniform architecture is not appropriate for more complex domains where chunking computations and assigning appropriate time costs is non-trivial; the myopic assumptions that they make create a "meta-greedy" barrier for search; and they ignore the cost of metareasoning itself. We might find ways to break the meta-greedy barrier by considering *limited* sequences of computational actions, with what

would amount to a stripped down *meta-meta-level* for choosing the sequences to reason about.

Boddy and Dean (Boddy & Dean 1989; Boddy 1991; Boddy & Dean 1994) study meta-level rationality for agents that reason with anytime algorithms. Anytime algorithms have many useful properties that simplify meta-level reasoning, and Boddy and Dean are able to use off-line reasoning to provide bounded strategic optimality for restricted problem domains. The approach scales to reasoning about the best use of resources when an agent must respond to a known sequence of deterministic events in the world. Horvitz (Horvitz 1987) complements the work by providing an explicit consideration of the costs of computation. He decomposes the utility of a computation into the *object-level*, that is time-independent, and the *inference-level*, that reflects the time-value of a decision. Horvitz presents an example to a time-critical decision problem in the medical-domain.

The work by Zilberstein and Russell (Zilberstein 1993; Zilberstein & Russell 1996) represents the maturity of the concept of bounded-rationality within AI. Zilberstein builds an operational framework that solves the important problem of the composition of anytime modules. His approach to metareasoning is also largely off-line, although he does allow a tradeoff to be made between off-line compilation of a metareasoning strategy, and on-line adaptive control of reasoning through active monitoring. The results show that there is reason to expect that the general problem of robust real-time decision-making in complex systems can be handled in practice, at least within the relatively clean context of anytime algorithms with well-defined performance profiles.

A long term goal of bounded-rationality within AI, and one which is totally absent at the moment, is the need to introduce new types of reasoning, and in particular *learning*. The myopic heuristics used by Russell and Wefald would clearly assess learning as costly, since learning brings only long-term goals. The idea of an agent that can adapt to bounded-optimal behavior is an interesting notion, and would be a very valuable direction of future research. Then Kasparov really would have reason to worry ....

References

Agre, P. E., and Chapman, D. 1991. Pengi: An implementation of a theory of activity. In *Proc. 9th National Conference on Artificial Intelligence (AAAI-91)*, 268–272.

Boddy, M., and Dean, T. 1989. Solving time-dependent planning problems. In *Proc. 11th International Joint Conference on Artificial Intelligence (IJCAI-89)* (1989), 979–984.

Boddy, M., and Dean, T. L. 1994. Deliberation scheduling for problem solving in time-constrained environments. *Artificial Intelligence* 67:245–285.

Boddy, M. 1991. Solving time-dependent problems: A decision theoretic approach to planning in dynamic environments. Technical Report CS-91-06, Department of Computer Science, Brown University.

Brooks, R. A. 1986. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation* 2:14–23.

Chase, W. G., and Simon, H. A. 1973. Skill in chess. *American Scientist* 394–403.

Cooper, G. F. 1984. Nestor: A computer-based medical diagnostic aid that integrates causal and probabilistic knowledge. Technical Report STAN-CS-84-48, Computer Science Department, Stanford University.

Cooper, G. F. 1990. Probabilistic inference using belief networks is NP-hard. *Artificial Intelligence* 42:393–405.

Doyle, J. 1989. Rationality and its roles in reasoning. In *Proc. 7th National Conference on Artificial Intelligence (AAAI-89)*, 1093–1100.

Good, I. J. 1971. Twenty-seven principles of rationality. In Godambe, V. P., and Sprott, D. A., eds., *Foundations of Statistical Inference.*

Grass, J., and Zilberstein, S. 1996. Anytime algorithm development tools. *SIGART Bulleting* 7(2):20–27.

Hansen, E. A., and Zilberstein, S. 1996. Monitoring the progress of anytime problem-solving. In *Proc. 14th National Conference on Artificial Intelligence (AAAI-96)*, 1229–1234.

Horvitz, E. J., and Breese, J. S. 1990. Ideal partition of resources for metareasoning. Technical Report KSL-90-26, Knowledge Systems Laboratory, Stanford University.

Horvitz, E. J. 1987. Reasoning about beliefs and actions under computational resource constraints. In *Proc. 3rd AAAI Workshop on Uncertainty in Artificial Intelligence*, 429–444.

Howard, R. A. 1966. Information value theory. *IEEE Transactions on Systems Science and Cybernetics* SSC-2:22–26.

Korf, R. E. 1988. Real-time heuristic search: New results. In *Proc. 6th National Conference on Artificial Intelligence (AAAI-88)*, 139–144.

Lin, S., and Kernighan, B. W. 1973. An effective heuristic for the Traveling Salesman Problem. *Operations Research* 21:498–516.

McCarthy, J. 1958. Programs with common sense. In *Proc. of the Symp. Mechanization of Thought Processes.* HMSO.

Newell, A., and Simon, H. A. 1972. *Human Problem Solving.*

Newell, A. 1981. The knowledge level. *AI Magazine* 2:1–20.

Russell, S., and Wefald, E. 1989. On optimal game-tree search using rational meta-reasoning. In *Proc. 11th International Joint Conference on Artificial Intelligence (IJCAI-89)* (1989), 334–340.

Russell, S., and Wefald, E. 1991. Principles of metareasoning. *Artificial Intelligence* 49:361–395.

Russell, S. J.; Subramanian, D.; and Parr, R. 1993. Provably bounded optimal agents. In *Proc. 13th International Joint Conference on Artificial Intelligence (IJCAI-93)*, 338–344.

Russell, S. 1995. Ratonality and intelligence. In *Proc. 14th International Joint Conference on Artificial Intelligence (IJCAI-95).*

Schoppers, M. J. 1987. Universal plans for reactive robots in unpredictable environments. In *Proc. 10th International Joint Conference on Artificial Intelligence (IJCAI-87)*, 1039–1046.

Simon, H. A. 1976. From substantive to procedural rationality. In Latsis, S. J., ed., *Method and Appraisal in Economics.* Cambridge University Press. 129–148.

Simon, H. A. 1978. On how to decide what to do. *The Bell Journal of Economics* 9(2):494–507.

Zilberstein, S., and Russell, S. 1996. Optimal composition of real-time systems. *Artificial Intelligence* 82:181–213.

Zilberstein, S. 1993. *Operational Rationality through compilation of anytime algorithms.* Ph.D. Dissertation, Computer Science Division, University of California, Berkeley, CA.